(2)

DataViews: A Personalizable, Interactive, Data Display System for the Decisionmaker

Report No. 87-10-1

Final Technical Report
ONR Contract N00014-83-C-0495
Work Unit NR DSA-009

Alan C Morse
Elizabeth Kohler

V.I. Corporation
160 Old Farm Road
Amherst, MA 01002

Jefferson Koonce
Nancy Tanner

Department of Industrial Psychology
and Operations Research
University of Massachusetts
Amherst, MA 01003

Date of report: 1 October 1987

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) 87-10-1	5. MONITORING ORGANIZATION REPORT NUMBER(S) Same
6a. NAME OF PERFORMING ORGANIZATION V.I. Corporation / 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research
6c. ADDRESS (City, State, and ZIP Code) 160 Old Farm Road Amherst, MA 01002	7b. ADDRESS (City, State, and ZIP Code) 800 N. Quincy Street Arlington, VA 22217-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Office of Naval Research / 8b. OFFICE SYMBOL (If applicable) Code 1142PS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-83-C-0495
8c. ADDRESS (City, State, and ZIP Code) 800 N. Quincy Street Arlington, VA 22217-5000	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. 65502; PROJECT NO. R1868801; TASK NO. R1868801; WORK UNIT ACCESSION NO. NR DSA-009

11. TITLE (Include Security Classification)
(U) DataViews: A personalizable, interactive, data display system for the decisionmaker

12. PERSONAL AUTHOR(S)
Morse, A. C., Kohler, E., Koonce, J., & Tanner, N.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final Report	FROM 83-07-01 TO 85-07-31	87-10-01	63

16. SUPPLEMENTARY NOTATION
Phase II of the DoD Small Business Advanced Technology Program

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Graphic display; Personalized display; Knowledge-based system; Flexible display system; Graphics workstation; DataViews.

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The development of a highly interactive, personal data exploration system (DataViews) is described. The system started as a subroutine library for creating dynamic graphs of data. Using a menu specification system, an interface was developed. This interface evolved thru three versions with the help of user feedback, advice from human-factor experts, and our own experiments, into an easy to use and powerful system. In addition, an experiment was devised to demonstrate the value of the system as a tool for human-factors experimentation. In that experiment, we used the system to compare three data-display techniques (analog coding, digital coding, and pitctorial encoding) in conjunction with redundant color coding.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL John J. O'Hare	22b. TELEPHONE (Include Area Code) (202) 696-4502 / 22c. OFFICE SYMBOL Code 1142PS

Table of Contents

List of Figures

List of Tables

Introduction

Think of the myriad decisions you make every day. How do you do it?

You **guess**.

But they are **educated** guesses, based on experience. You evaluate data from various sources to extrapolate the consequences of each alternative. Then you pick one. Often you are not aware of this data evaluation, so let's use an example to make it more concrete.

Imagine you are at a buffet deciding what to eat. How are you going to choose? You might start by classifying the foods (measuring them using a nominal scale): appetizers, entrees, desserts, foods you like, foods you should eat but aren't crazy about, foods you should avoid, and foods you dislike but will take to be polite. Next you might rank the foods (measure them on an ordinal scale): foods you will eat first, foods you like best. Finally, you might determine amounts (measuring data on an interval or a ratio scale): how much time do you have to make up your mind, how long will you have to wait before you can get in line again, how much of each item should you take.

Thus: **Decisions are based on measurement.**

But measurement is more than just assigning a number to an object. Recall the first time you had to shop using a foreign currency. The measurements of value were explicit: this tube of toothpaste costs 1000 zlotys. Should you buy it? Suppose there was another brand of toothpaste next to it that cost 99 zlotys. Would you buy it then? Or suppose you had bought a loaf of bread the day before, and it had cost 2 zlotys; now, would you buy the toothpaste?

Measurement is comparison. It involves comparing things with other things and comparing things with experiences.

Our gut-level manner of dealing with the world involves making internal analogs of external entities. Thus, we deal with the world **directly** in an analog fashion. In our imaginary buffet, when we fill our plate with mashed potatoes, we are weighing the food by making an analogy of weight to kinesthetic measures of muscle and ligament tension. Moreover, a measure is never separate from the object being measured, because we need the object to give the measure meaning. A pound of feathers weighs the same as a pound of lead, but we don't think of them as the same.

But how do we **communicate** our experiences of the world? Let's take as an example how we experience time. We experience it directly as an analog quantity, but we communicate it to others in a digital form, because it is more compact, universal, and precise. When we receive experiences digitally, we are experiencing the phenomenon **indirectly**. To comprehend digital information we often transform it to a more directly accessible form, namely into some internal, personal analog quantity. This is why graphs are so effective. They help us translate digital values into analog form.

Thus, we deal with the world using intuitive, analog comparisons and learned digital comparisons.

In making decisions, we use both the direct and the indirect measures of the world. In the latter case, making decisions also requires that we translate the information into a more personally-accessible analog form. We see this when we try to analyze a table of numbers, where we may find ourselves comparing the numbers by comparing the length of the digital representations. (This is equivalent to a bar graph of the log of the data.)

In sum, decision-making involves interpreting and comparing: 1) Direct experiences, which are analog, personal, intuitive, imprecise; and 2) Indirect experiences, which are digital (symbolic), transpersonal, communicable, precise, and which require translation into some personal form.

To facilitate this process, mathematicians and statisticians have evolved standard graphical data representations, like tables, line graphs, bar charts, and pie charts. (EHR75, SCHM79, and TUFT83 contain excellent descriptions on the relevant graphical design issues.) These techniques evolved in a pencil and paper environment, and are suitable for manual implementation. With graphical workstations coming into widespread use, these techniques have been automated and expanded. But there is a great opportunity to expand on these graphical data analysis techniques.

This is the motivation behind the system we developed, DataViews. It came from our desire to create a data exploration system, which takes digital data and transforms it into personally meaningful animated pictures, using an easy-to-use, highly interactive, menu interface. (See MORS82 and MORS84.) Similar systems that have been develop independently by others include STEAMER (HOLL83, STEV83), ModVUE (GOUL82), and PVS (FOLE86).

The source of the digital information for DataViews is multifarious; it may come from data-acquisition devices monitoring experiments or processes, or it may come from simulations. The typical user is an engineer, who might not be a programmer. The ideal hardware environment is a medium-to-high resolution color graphics workstation with a mouse as a graphics input device.

In this paper, we describe:

- The Menu Specification Language, used to define the interface to DataViews.
- DataViews: its features, how it was evaluated, and how it evolved.
- An experiment that evaluated the DataViews interface.
- An experiment that used DataViews to examine one of its basic premises, namely, that suitable graphical data representations can enhance performance in a data monitoring task.

Menu Specification Language

In order to expedite interface development and modification we developed a menu specification language (MSL), which we use to specify menu interactions for DataViews. (See Appendix A for detailed description of the MSL and an example menu specification.) Using the MSL requires that programmers write menu descriptions, which descriptions are interpreted at runtime by the Data-Views menu interpreter. Menu specifications may be compiled into a more compact intermediate form before being interpreted by the Menu Interpreter at runtime. This reduces the startup time when the menu description is read in. This approach allows some redefinition of the interface without recompiling the DataViews system.

The MSL has the following features:

- It encourages modular, structured implementation.
- It promotes an "end-user" view of interface design, giving the ease of **using** a program a higher priority than the ease of **writing** the code for the program. Developers therefore plan interfaces based primarily on users' needs.
- It is device-independent, enabling programmers to develop interfaces for systems with different display and cursor control capabilities.
- It can call subroutines in the program, provided their names have been added to a special symbol table.

The MSL comprises three primary entities;

- **Variables**, analogous to scalar variables in a programming language.
- **Actions**, like subroutines, are composed from a set of primitives actions, variables, and control structures.
- **Menus**, which is a viewport containing text or graphics that identifies an action to be taken when the user points to the text or graphic entity.

These components are described in more detail below.

Variables

MSL variables can be one of the following types:

- **Simple variables** of type: CHAR, SHORT, INT, LONG, FLOAT, and DOUBLE, which correspond to the C language types, and which may be arrays.

- **Strings**, which are one-dimensional null terminated CHAR (character) arrays set to an initial value.

- **Externally** referenced **variables**, defined in user-written C subroutines.

- **Viewports**, which are arrays of four SHORTs that define the lower left and upper right corners of a rectangle on a screen.

- **Viewport attributes**: foreground and background colors (in RGB format) and associated text attributes (size and font).

- **Viewport lists**, which are arrays of viewports with their attributes.

- **Key bindings**, which associate keys with actions, such as the '?' key associated with a help action.

Actions

Actions are like C functions in that they are sequences of primitive statements, which include:

- **Control structures**: IF-THEN-ELSE, WHILE-ENDWHILE, and REPEAT-UNTIL.

- **Subroutine calls**, for example, user-supplied functions written in C.

- **Action invocations**.

- **Menu invocations**.

Menus

Menus are like modules in that they are collections of related actions that become accessible as a group. Thus, when a menu is called, it is displayed by the menu interpreter until some action is performed that causes it to disappear. Menus contain the following items:

- Actions to be taken before the menu is drawn.

- Menu attributes: color, shape, position, and graphics or text.

- Menu items, which are the menu selections. These usually contain text or graphics that identify the choice, and action that is to be performed when the item is chosen, and an associated help message.

- Actions to be taken after the menu is exited.

DataViews

The DataViews system actually comprises several components:

> **DV-Graph,** the initial version of DataViews, this was a menu-based graphing package.
>
> **DV-Draw**, the improved, "evolved" version of DV-Graph, containing additional drawing capabilities.
>
> **The Menu Specification Language**, the in-house tool for specifying menu-based graphical interfaces. (See above.)
>
> **DV-Routines**, which is a subroutine package providing programming access to some of the capabilities provided in DV-Draw.

Decisionmakers would most likely use the DV-Draw component of DataViews as a data analysis aid.

DV-Draw, then, allows users to combine graphs with drawings. The drawing elements (for example, lines, rectangles) may have dynamic attributes (color). This is useful for creating schematics of processes and attaching dynamics and graphs to the schematic. An obvious application is in instrumentation and process monitoring. (See Figures 1, 2, and 3 for examples.) MORS84 describes the functionality of DataViews and a command-based drawing editor, compares them to related systems, and provides several examples of their use. DV-Draw is a menu-based version of the drawing editor, and it contains many of the features described for that system. Since it differs, slightly, we summarize the current capabilities of DV-Draw below.

Capabilities

DV-Draw combines a simple computer-aided drawing system with a graphing package. It allows users to create dynamic drawings, comprising graphs, lines, text, and (possibly filled) rectangles, circles, arcs, and cubic b-splines. The dynamic elements can be linked to user-defined variables, which drive the dynamics. Moreover, a drawing can contain references to other drawings, so that users can define their own symbols. Once created, these drawings can be "run" in order to display the data, or they can be saved and "played back" by a special program that runs them from the operating system.

These drawings have three basic types of dynamics:

> **Graphs**, which come in a variety of flavors, for example, bar chart, line graph, dial, dial with history, 3d surface plot, size encoding.

Color thresholding, which causes the color of graphical objects to change as a function of data value. See Figure 4 for a schematic of a color threshold table in which the colors have been encoded as patterns. The table has input data values, and it outputs the color that the graphical object is to be displayed in. It determines the color using a user-defined table of threshold-color pairs. This mechanism works for all drawing entities except graphs.

Drawing thresholding is similar to color thresholding except its output is a drawing rather than a color. Figure 5 shows a schematic for a drawing threshold table which outputs a drawing of an open valve or a closed valve, depending on the input data value. This type of dynamics can only be applied to subdrawings (references to other drawings within a drawing).

Interface Principles

The principles used to define the user interface are:

Object-oriented. The objects (circles, lines, arcs, etc.) in the drawing define the actions that are appropriate for them. This leads to the following point:

Object-verb(vs. verb-object) command structure. The user generally interacts with the drawing by first selecting objects (circles, lines, arcs) in the drawing and then choosing actions (verbs) that are appropriate for those objects. These actions appear in a menu when the objects are selected.

Immediacy of effect. Every menu selection should result in some immediate, visible effect.

Flat menu structure. The tree of menus should be as flat as possible, minimizing disorienting context switches, and maximizing random access to different parts of the tree.

Task Orientation. The interface and data structures should derive from the task. This means that simple tasks should be simple to perform. This also means data structures should be created with reasonable default values.

Self evidency. Each step should be self-evident, without requiring a complicated model of the system. (This principle follows from Immediacy of effect and Task orientation.)

These principles described were derived from the human factors literature, and from our experience in implementing several versions of the interface.

Evolution of the DV-Draw Interface

The DataViews system went through three interface versions, driven by user feedback (from customers), expert evaluation (by consultants), and a human factors experiment (described below). In this process, DV-Graph evolved into DV-Draw and we precipitated the principles described in the previous section. The major flaws that were discovered in the first interface derived from the historical context.

Evolution: Phase I

DataViews started as a subroutine graphing package that was to be sold without an interactive interface, and its structure was defined with that in mind. When we added the interface, we did it naively; we created what amounted to a menu-based subroutine invoker (the first version of DV-Graph). This resulted in an interface that reflected the underlying data structures, rather than the data structures reflecting the interface. This required users to develop a model of the data structures, which had a structure that was not entirely relevant to their task.

So the above principles were violated. That is, the first system was:

Action-oriented. The menu choices matched the subroutine calls.

Verb-object. The user generally specified the action and then the data to which it was to be applied.

Without immediacy of effect. Because data structures were being modified, there wasn't always an obvious result that could act as user feedback.

Deeply hierarchical. The tree of menus reflected the deep hierarchy of the data structures.

Data structure orientated. The interface derived from the data structures, not the task.

Without self evidency. The system required that the users have a model of its behavior before they could start using it. (Rather than the system having a model of the user's behavior.)

Since most users did not have the model of the data structures that we had as designers of the system, they found it error-prone, difficult to learn, and cumbersome.

We discovered these problems pretty early, both as we used the system and as we watched others use it. So we set about trying to improve the system. The steps involved are described in the following section.

Evolution: Phase II

To help improve the interface, we hired a human factors consulting firm, Human Performance Associates, to give us some guidance. They suggested some of the principles mentioned above and worked with us as we started to implement them. The ones we focused on initially were:

Task orientation. We tried to create a reasonable user model of the system. How would a user think of a system that was supposed to allow simple, flexible display of the data? Where was the data to be found? As part of this task orientation, we made sure that all data structures had some defaults (as reasonable as we could make them).

Immediacy of effect. For every data structure on the system, we needed to have some visible counterpart, which would change when the user modified its value. This also meant that every action should only take a few steps to complete (worst case: 3 steps; best case: 1 step).

Self evidency. If the system was properly task oriented, then self evidency should follow. It would require that the interface provide enough hints as to the acceptable range of actions.

The result of this analysis was an improved, but not entirely satisfactory interface. So we tried again.

Evolution: Phase III

This time we tried experimental evaluation. We hired Neil Hirsch Associates to experimentally evaluate the interface. They observed several subjects using the system and gave us a list of recommendations (HIRS84), some of which are included in Appendix B. In addition, we performed our own experiments on the system with the help of members of the Department of Industrial Psychology and Operations Research at the University of Massachusetts. (This experiment is described in the next section.) Finally, we explored several similar products on the Apple Macintosh, which we felt had successfully incorporated the principles described above.

We then used this experience in defining the interface to DV-Draw, which added CAD-like utilities to the graphing functions. Finally, we had something we thought was satisfactory. Figure 6 shows a before and after schematics of the menu trees for DV-Graph (which had a similar interface to DV-Draw), showing how the menu structure became flatter. Figure 7 shows the top level menu of DV-Draw.

Of all the approaches taken to improve the system, the most effective was comparing it to similar systems on the Macintosh that had good interfaces. This caused an epiphany, "So that's what it should look like. It's obvious once you see it. The difference in the product was dramatic. Current customers find little need to refer to the manual, and they are able to use the system with minimal instruction.

Experiment 1: DataViews Evaluation

As described in the previous section, DataViews had three major releases, each with substantially different (and improved) interfaces. Before moving from release 2 to release 3, we performed experiments on the system with the help of members of the Department of Industrial Psychology and Operations Research at the University of Massachusetts. This experiment is described below. We then used this experience in defining the interface to DV-Draw, the final version of the DataViews editor.

Purpose

Our main purpose in performing this experiment was to evaluate version 2 of DataViews, both in determining where users tended to determine weaknesses in the interface and to determine subjective ease-of-use in performing a data display task. We also wished to determine those concepts in DataViews that users would be likely to have difficulty understanding.

To this end, we chose a task that we felt was representative of how the system would be used. We asked the subjects to create a display that was similar to a car's dashboard.

Method

A total of thirty-six paid volunteer subjects between the ages of 18 and 29 were brought to VIC for the purpose of evaluating the DataViews system. The running of the subjects was performed by two graduate research assistants from the Department of Industrial Engineering and Operations Research, University of Massachusetts. The subjects were students in engineering or computer sciences, and all had some experience in working with computers. Each subject was paid ten dollars at the end of his/her research session (generally less than two hours).

The experimenters explained the task and had the subjects sign a consent form acknowledging understanding of the purpose of the study, the disposition of the data, and their rights to discontinue participation as a paid volunteer.

The research session was divided into two separate phases. The first used fifty numbered cards of instructions to lead the subject through the learning of how to use DataViews to create graphic displays. After completion of the first phase, the subjects took a break (three to five minutes) and then began Phase 2, in which they were to develop a pre-determined display (see Figure 8 using the techniques covered in Phase 1.

The instruction cards of Phase 1 were available throughout Phase 2 in case the subject felt a need to refer back to how a particular step was performed previously.

Eight subjects were deleted from the final data analysis for the following reasons:

- One subject had a total system failure ("crash") during Phase 2. This subject finally accomplished Phase 2, but the additional experience in the "re-do" effort affected the time to progress through Phase 2.

- Two subjects did not work with the standard system during their training and testing. The standard system was not available and a backup systems was tried. The differences between the systems turned out to significantly affect the performance of the subjects.

- Five subjects failed to complete Phase 2 within the alloted time of two hours or had personal problems that would invalidate their Phase 2 data.

Information regarding particular sub-tasks that gave the subjects difficulty, problems in using various features of the system, and subjective comments by the subjects and observations of the experimenters were collected and collated for all subjects throughout the entire time of Phase 1 and Phase 2. In addition, the dependent variables included the time to through the familiarization instructions (Phase 1), the time to create the designated display (Phase 2), and the numbers of the cards referred to during Phase 2.

Results

A comparison of the two experimenters was made to see if there were any experimenter biases affecting the subjects' performances. With respect to the time to perform Phase 1, the time to perform Phase 2, and the number of instructions referred to by the subject during Phase 2, there were no differences between the two experimenters.

The mean age of the subjects was 22.23 years with a standard deviation of 3.03 years, and there were sixteen males and fifteen females who completed the questionnaire at the end of the experiment.

With respect to the performance timings, the females took significantly less time than the males ($p<0.02$) to complete Phase 1 (mean (F) = 38.44 min., mean (M) = 49.63 min.) and there was no significant difference between sexes in completing Phase 2 (mean (F) = 62.10 min., mean (M) = 62.17 min.).

After performing the required tasks, the subjects were asked to complete a brief questionnaire (Appendix C) regarding their personal experiences with computer systems and the research in which they had just participated. The responses to questions 1 through 4 are plotted in Figures 9 through 12. The respondents were put into three different categories with respect to the amount of experience they reported to have had with computers; Group I - very little (n=5), Group II - moderate (n=9); and Group III - considerable (n=20).

Subjects generally felt that they had sufficient time to perform the required tasks; that is, they did not feel rushed. The Group I subjects did not feel that too much time was given, an expected response for computer-naive subjects.

Surprisingly, the more experience the subjects had, the less they liked the mouse system. All three groups were generally favorable towards the mouse, but Groups II and III were less favorable than Group I. In fact, the inexperienced subjects were rather impressed with it.

The respondents in the three groups were about equal in their attitudes towards the menu system. Overall, they liked it and found it easy to use.

Answers to the last question did not show any particular attitude about the utility of the instruction cards. The only unique aspect of the results was that all the subjects in Group I responded the same, a rating of 4.

Kruskal-Wallis tests yielded no significant differences among the responses by the three groups.

With regard to the subjects' previous computer experiences, analyses were performed on the time to completion for Phases 1 and 2. There were no significant differences among the three groups in Phase 1 (means of 37.9, 43.9, and 45.0 min. for the three groups, respectively) and in Phase 2 (means of 67.1, 65.6, and 56.0 min.). Contrary to expectations, the more experienced subjects took longer to learn the system (Phase 1), but took less time to construct the pre-determined display (Phase 2).

Looking at the amount of time to perform the Phase 2 task minus the time it took to perform the Phase 1 task, no significant difference was found among the groups ($p<0.10$, with mean differences of 29.0, 21.7, and 12.7 min.). Figure 13 shows the relationship of the time to learn the system (Phase 1) to the additional time required to create the display (Phase 2). From this, we can see that there is a tendency for subjects who took less time in Phase 1 to take longer in Phase 2, and vice versa. The actual correlation between time in Phase 1 and the additional time required for Phase 2 was $r=-0.675$, $p<0.001$.

Appendix B summarizes the experimenters' evaluations of the DataViews system and how the comments affected the design of DV-Draw and version 3 of DataViews.

Experiment 2: Using DataViews to Evaluate Data Display Formats

In this section we describe an experiment that evaluated the effectiveness of various data display formats. We performed the experiment for two reasons: to look at one of the fundamental premises that motivated the creation of DataViews (namely, that graphical data representations should enhance performance as compared to digital data representations); and to assess the value of DataViews as a tool for human factors experimentation.

Purpose

Given a process control context, where a person is required to monitor and control several components of a complex system, we asked, "What is the best way of displaying the important information about the state of the process?" We considered two aspects of the display:

- the form in which the information is displayed (digital, analog, or pictorial) and
- whether the information is presented redundantly using some code other than numerical coding (like, color or shape coding).

In this particular series of studies, we examined the role of color coding in facilitating a data monitoring task. There were three conditions: 1) no color coding; 2) 2 colors, where the color changed at significant thresholds; and 3) 3+ colors, where color changed at significant thresholds and as the data approached significant thresholds. CAHI76 found that the effect of color coding on reaction time in a search task was best described by a U-shaped function, where the addition of a coding with a small number of colors improved performance, but the addition of many colors degraded performance. These effects can be quit task specific, however (CHRI75).

In additions to examining the effects of redundant color coding, the efficiency of three different types of displays (digital, analog, and pictorial) was compared. As we discussed in the introduction, there are intuitive reasons to expect an advantage of a graphic or analog display over a digital display.

The dependent variables were: 1) response time, 2) percent hits, and 3) false alarms. The task comprised monitoring 5 gauges, checking for threshold crossings and comparison of two instruments.

Method

Fifty-four paid volunteers between the ages of 18 and 29 came to VIC for testing in a room that was isolated from the rest of the VIC activities. The running of the subjects was performed by a graduate research assistant from the Department of Industrial Engineering and Operations Research, University of Massachusetts. The subjects were students in engineering or computer sciences, and all had some experience in working with computers. Each subject was paid ten dollars at the end of his/her research session (generally less than two hours).

The experimenters explained the task and had the subjects sign a consent form acknowledging understanding of the purpose of the study, the disposition of the data, and their rights to discontinue participation as a paid volunteer.

The research session was divided into four ten-minute runs. A Latin Square Design was used to apportion the different display conditions among the subjects. (See Figure 14) Each subject was exposed to each format (digital, analog, and pictorial), and a final display. (Examples of each of the display types are shown in Figures 15, 16, and 17.) Half the subjects were able to select the display format for the final display, while the other half had it preselected. The different formats were presented with different event frequencies.

Subjects were to hit a key as soon as they detected a critical event, and the response time was recorded by the computer. The critical events were as follows.

- **Oil pressure too high**. Pressure higher than a fixed threshold.
- **Oil pressure too low**. Pressure lower than a fixed threshold.
- **Fuel level too low**. Fuel lower than a fixed threshold.
- **Speed too high**. Speed higher than a varying threshold, which was displayed in another graph.
- **Water temperature too high**. Pressure higher than a fixed threshold.

In addition to varying the type of display, three separate groups of 18 subjects each experienced different amounts of redundant color information. For the non-color coded group, the color of the gauges remained constant, regardless of the values shown. The binary color coded group viewed displays where the components changed color depending on whether the values were above or below the critical threshold. (Since the critical threshold varied in the speed case, the color change in the speed display changed at a fixed value in the middle of the range of varying thresholds.) Finally, for a third group of subjects, color was associated with the values of the monitored display components. Three to four colors were used, where save values were represented in green, critical values in red, and intermediate values in one or two shades of yellow.

Only half of the events were critical. Non-critical events were jumps in the data displayed towards the threshold but not crossing it. This allowed us to test how well subjects were able to discriminate for each display condition.

The experimenter explained the criteria for critical events and gave an example of each. Then the experimenter started the run using the appropriate display format type. The computer recorded the response times and saved them for later analysis.

Three runs (subject 4, run #1; subject 18 runs #1 and #4) were deleted from the final data analysis because the data files were inadvertently corrupted. In the analysis, these data were replaced by the average of the data for the other subjects who had the identical conditions.

After performing the required tasks, the subjects were asked to complete a brief questionnaire (Appendix E) regarding their preferences for the display types, and their assessment of the task difficulty.

Results

The mean age of the subjects was 26.5 years with a standard deviation of 5.1 years, and there were thirty-one males and twenty-three females who completed all four runs.

A 3x3x3 mixed design Analysis of Variance was performed on each of the three behavioral measures (reaction time (RT), percent hits, and false alarms) for the first three runs. Further comparisons between means were made using the Duncan Multiple Range test.

The mean of median RTs, averaged across all three runs, are shown in Figure 18. As expected, the association of color with the threshold improved performance (F=43.05; d.f.=2,49; p<.001).

The association of two colors with the threshold produced the fastest RTs while a lack of color coding produced the slowest RTs (DMR, p<.01). The multi-color coded group had RTs significantly slower that the binary coded group but faster than the non color coded group (DMR, p<.01). The fact that the use of more than two colors degraded performance is consistent with the findings of CAH176.

The effect of display type was also significant (F=57.90; d.f.=2,113; p<.001). The results of the Duncan Multiple Range Tests showed that performance with a digital display was significantly worse than with either a pictorial or analog display (p<.01); however, no significant difference was found between the use of an analog or pictorial display. This is consistent with the findings of TULL81.

A significant display by color code interaction (F=30.29; d.f.=4,113; p<.001) probably occurred because of the very positive effect of the inclusion of color codes on the digital display (DMR, p<.01) and because of the greater sensitivity to display type of the noncolor coded group (DMR, p<.01). The latter may have been the result of a floor effect. While the non-color coded group showed the most sensitivity to display type, display type also affected the multicolor coded group. That is, there was a significant decrease in RT for this group with the use of the analog or pictorial display rather than the digital display . Display type had no effect on the RTs of the binary color coded group (floor effect).

The interaction of display type with run number is shown in Figure 19 (F=4.05; d.f.=2,49; p<.025). This interaction was a result of an increase in RTs with the digital display across runs. Neither the analog nor the pictorial display showed this effect. Caution must be used in interpreting this as the result of fatigue since the data for a digital display on run number three is based on a different subject set than on run 2 or 1. It is interesting to note that, in general, there were no practice or fatigue effects (i.e., the overall effect of run number was not significant).

The reaction time measure reflects the speed of the subjects' response. The two other behavioral measures studied, percent hits and false alarms, reflect the accuracy of the subjects' response. The mean percent hits averaged across the three runs are shown in Figure 20. The three different levels of color coding did significantly influence the percentage of hits (F=46.42; d.f.=2,49; p<.001). Duncan's Multiple Range test revealed that the two groups with color coding performed better than the non-color coded group (p<.01). No difference was found between the use of two colors or multiple colors for coding.

The effect of type of display on the percentage of hits was significant (F=27.20; d.f.=2,113; p<.001). The digital display produced the smallest percentage of hits, followed by the analog display with the pictorial display producing the largest percentage of hits (DMR, p<.01). Note that, whereas there was no difference in speed of response to the analog versus the pictorial display, the detection of critical events was significantly greater with a pictorial display rather than an analog display.

The interaction of type of display with the level of color coding was significant (F=26.97; d.f.=4,113; p<.001). This interaction takes the same form as the same interaction analysis of the reaction time measure. That is, there was a greater effect of level of color coding on performance with a digital display than on the other two display types and the non-color coded group showed greater sensitivity to display type than did the other two color coding groups (ceiling effect). Duncan's Multiple Range test showed that performance differed significantly in the non-color coded group with the three display types (p<.01). That is, performance was best with a pictorial display followed by an analog display and worst with the digital display. This effect was not seen in the other two color coding groups. Furthermore, there was a highly significant effect of level of color coding on the digital display (p<.01) but less of an effect on the analog display (p<.05) and no effect on the pictorial display (p>.05).

The other measure of accuracy observed was the mean number of false alarms (Figure 21). A false alarm occurs when a subject responds and there is no critical event. Once again, the level of color coding did affect performance (F= 6.99; d.f.=2,49; p<.005). It is clear from the figure that the multicolor coded group made many more false alarms. Indeed, the Duncan's Multiple Range test indicated that the multicolored coded group made significantly more false alarms than either of the other two groups (p<.01) while there was no significant difference between the other two groups. It is interesting to note that the other two behavioral measures (percent hits and RT) indicated that performance was worse with no color coding and best with either binary or multi color coding. However the error rate, as represented by false alarms, was highest for the multiple color coded group. The problem probably lies in the fact that there were occasional dramatic shifts in color for this group that were not associated with a critical event.

Unlike the other two behavioral measures, the number of false alarms was not influenced by the type of display (F=1.71; d.f.=2,113; p>.05). However, this was the only measure which showed an effect of run number (F=3.06; d.f.=5,113; p<.025). That is, the mean number of false alarms decreased on each successive run (run 1 mean=4.69; run 2 mean=3.04; run 3 mean=2.57). This decrease was significant from run 1 to run 2 and from run 1 to run 3 but not from run 2 to run 3. That is, the improvement occurred from run 1 to run 2 and no further improvement occurred.

The display type by level of color coding interaction was not significant for false alarms. There was, however, a significant interaction of display type with run number (F=4.19; d.f.=2,49; p<.025). This interaction is shown graphically in Figure 22. It appears to occur because the greatest decrease in false alarms occurs on run 2 for both the analog and digital display but not until the third run for the pictorial group. Perhaps this is again because of the greater tendency in the pictorial display for radical color shifts to occur independently of a critical event. The fact that all three display types produce about the same number of false alarms by run 3 probably indicates a floor effect.

After performing the required tasks, the subjects were asked to complete a brief questionnaire (Appendix D) regarding their preferences for the display types, and their assessment of the task difficulty. The results are summarized in Table 1.

Question 1 asked subjects to select the display which they found the easiest to monitor. Of the 54 subjects, 26 found the pictorial display to be the easiest to use, 16 found the analog easiest, and 12 found the digital the easiest. While a chi square did not indicate that these choices were significantly different, the chi square was marginally significant (chi square = 5.365, df = 2, .10 > p > .05) indicating a tendency for subjects to favor the pictorial display. A 2-way chi square performed on the data for all three groups separately indicated that the 3 groups did differ significantly in their preferences (chi square = 10.49, df = 4, p < .05). This was a result of the fact that the non-color coded group showed a stronger preference for the pictorial display and a stronger distaste for the digital than did the other two groups. This pattern of result is the same as that found for both RT and % hits, where it was seen that the non-color coded group was more sensitive to difference in display type than were the other two color coding groups.

The second question was very similar to the first; it asked subjects which display type they liked best. It was felt that subjects might respond differently to this than they did to question 1, because the might view the pictorial display as more interesting or more "fun," even if it was not the easiest to use. Indeed, subjects did show a strong tendency to favor the pictorial display (chi square = 10.825, df = 2, p < .01). A 2-way chi square performed on all three groups showed that there was no significant difference between the groups in their preferences for display types (chi square = 7.509; df = 4; p > .10). This result reflects the fact that the tendency to strongly favor the pictorial display was consistent across the 3 groups.

In question 3, subjects were asked to rate the difficulty of the task from extremely easy to extremely difficult on a 7 point scale. As can be seen from the table, the most frequent responses were in the middle range of the scale, indicating that the subjects found the task of moderate difficulty. A chi square performed on these ratings indicated that there were no differences between the three groups in terms of their reported perception of the task difficulty (chi square = 9.675; df = 12; $p > .05$)

Summary

The data indicate that the analog and pictorial displays are easier for subjects to monitor. These two display, types produced the fastest RTs and the best accuracy as defined by the percentage of hits. The pictorial display produced a slightly higher percentage of hits than the analog display. The number of false alarms was not affected by type of display. The addition of color coding clearly improved the subjects' ability to detect critical events quickly and accurately, however, the use of two colors for coding produced better performance than the use of more than two colors. The optimal display then in terms of rapid, accurate responding should be either analog or pictorial and should be coded in only two colors. (In terms of percent hits, the pictorial display has a slight advantage)

In performing this experiment, we found that DataViews was an effective tool, in that it allowed us to quickly create and evaluate displays; it made it easy to write a program that ran the displays and collected the subjects' responses. Thus, the time required to set up the experiment was minimal. Naturally, we chose an experiment that would be easy to implement with DataViews; but we feel that it is representative of a large number of useful experiments that could be performed.

Conclusions

In this paper, we have described the creation and evolution of a powerful tool for data display. It was motivated by our belief that effective understanding of data requires converting it to an intuitive, internal analog form. With powerful computer graphics workstations in widespread use, we are in the position to facilitate that conversion process. DataViews has taken a major step in that direction, primarily because of the time we took to evaluate the interface.

We also found that DataViews can be an effective tool in human factors experimentation, because of the ease in which prototype displays can be created. Encouraged by this result, we have donated copies to the University of Massachusetts in Amherst, where it is being used in the Department of Industrial Engineering and Operations Research (among others) for display prototyping and testing.

Currently, DV-Draw has an installed base of about 400 users, who verify that it is indeed easy to use. Our customer support department receives very few calls with questions about how to use it. This is the case despite the fact that most users only read the manuals in a cursory fashion, if at all.

Despite its ease of use, DV-Draw has limitations. Like any closed system, it cannot be all things to all people. Users will always have idiosyncratic requirements that have not been anticipated by the designers. With a product as general as DV-Draw, it would be impossible to fulfill all these requirements. Therefore, we found it necessary to open up DV-Draw by providing a subroutine interface, which we call DV-Tools. This interface is at a much higher level than the DV-Routines interface mentioned earlier. It allows integrating drawings created using DV-Draw into applications, which use them as dynamic data displays. This has the effect of greatly facilitating interface prototyping and development for real systems. It wasn't until we did this that DataViews became a viable product in the marketplace.

The prospects of DataViews are very promising, and there are many directions it can take. In particular, we are looking at:

- enhancing its capabilities as an interface design system by creating a standalone interface prototyping system, which would require no programming by the user;
- integrating statistical functions to enhance it as a data analysis tool;
- adding an event monitor, which would act like an expert system that would monitor the incoming data for user-defined events;
- incorporating AI in the form of an expert system that would help users in the design of their displays.

References

BERT83 J. Bertin. **Semiology of Graphics**. The University of Wisconsin Press, Madison, 1983.

CAHI76 M. Cahill and R.C. Carter. Color code size for searching displays of different density. Human Factors, Vol. 18, No. 3, 1976, pp. 273-280.

CHRI75 R.E. Christ. Review and analysis of color coding research for visual displays. Human Factors, 1975, Vol. 17, pp. 542-570.

FOLE86 J.D. Foley and C.F. McMath. Dynamic Process Visualization. IEEE Computer Graphics & Applications, Vol. 6, No. 3, March 1986, pp. 16-25

EHRE75 A.S.C. Ehrenberg. Data Reduction: Analysing and Interpreting Statistical Data. John Wiley & Sons, New York, 1975.

GOUL82 Technical literature on the Modvue System, Gould Electronics, Andover, Mass., 1982.

HIRS84 DataViews: Analysis and Recommendations. Report from NH Associates, San Jose, CA. August, 1984.

HOLL84 J.D. Holian, et. al. STEAMER: an interactive inspectable simulation-based training system. **The AI Magazine**, 1984, V, 2.

MORS82 A.C. Morse, et. al. The development of an intelligent, trainable graphic display assistant for the decisionmaker. Intelligent Software Systems, Inc., Amherst, Mass., Report No. N00014-82-C-0136, May, 1982.

MORS84 A.C. Morse. A system for embedding data displays in graphical contexts. V. I. Corp., Amherst, Mass., Report No. 84-7-1, July, 1984.

PETE82 R.J. Petersen, et. al. Performance-based evaluation of graphic displays for nuclear poser plant control rooms. Human Factors in Computer Systems, Proceedings, March 15-17, 1982, pp. 182-189.

SCHM79 C.F. Schmid and S.E. Schmid. **Handbook of Graphic Representation (2nd ed.)**. John Wiley & Sons, New York, 1979.

STEV83 A. Stevens, et. al. The use of a sophisticated graphics interface in computer-graphics assisted instruction. IEEE Computer Graphics & Applications, Vol. 3, No. 2, March/April 1983, pp. 25-31.

TUFT83 E.R. Tufte. **The Visual Display of Quantitative Information**. Graphics Press, Cheshire, Connecticut, 1983.

TULL81 T.S. Tullis. An evaluation of alphanumeric, graphic, and color information displays. Human Factors, Vol. 23, No. 5, 1981, pp. 541-550.

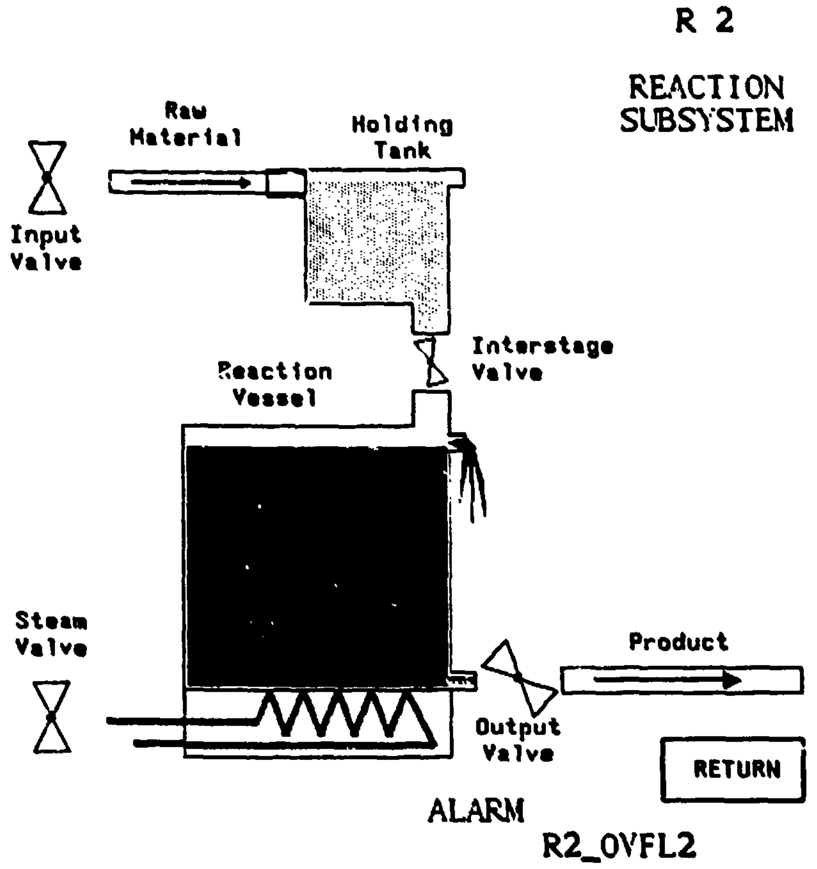


Figure 1. Example DataViews Application. This shows a portion of a chemical processing tank. Fluid enters the holding tank at the top, as controlled by Input Valve; it flows to the reaction vessel, through the interstage valve; and it exits the system through the output valve. The temperature of the reaction vessel is controlled by the steam valve. In the application, the user uses the mouse to open and close the valves and to request additional information. For example, if the user selects the word "Product" a strip chart appears showing the amount of output that has been produced. If the user points to a tank, a new display appears, showing several graphs describing the state of the tank.

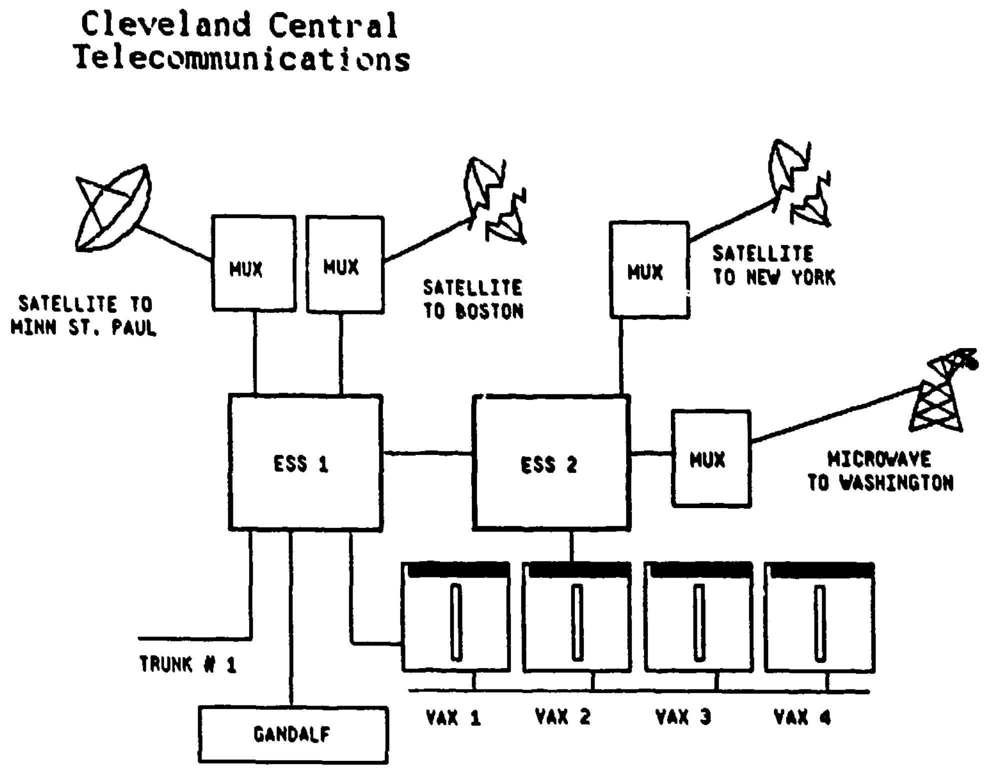


Figure 2. Example DataViews Application. This shows a portion of a telecommunications network. It contains both color and drawing dynamics. The color dynamics is manifested in the links between the components, which links change color according to the activity on those links. The drawing dynamics is manifested in the satellite and microwave towers, which switch between the "broken" icon and the normal icon. In addition, users can request graphs of activity by picking the desired object.

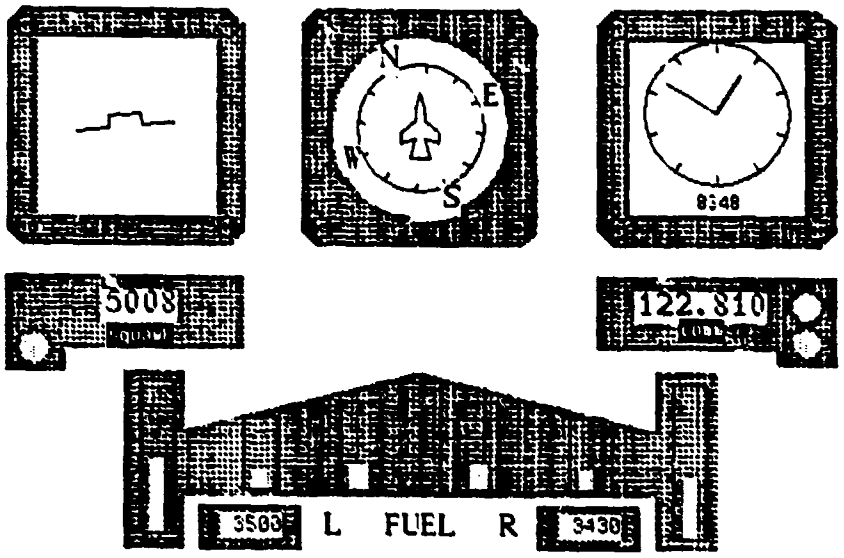


Figure 3. Example DataViews Application. This shows a simplified airplane cockpit, which was created using DV-Draw in about thirty minutes. The top three instruments are (from left to right) an artificial horizon, heading, and altimeter. The bottom portion of the display is a schematic of the fuel tanks contained in the plane's wings. The bars indicate the amount of fuel in each of the size wing tanks. As fuel is consumed, they become shorter.

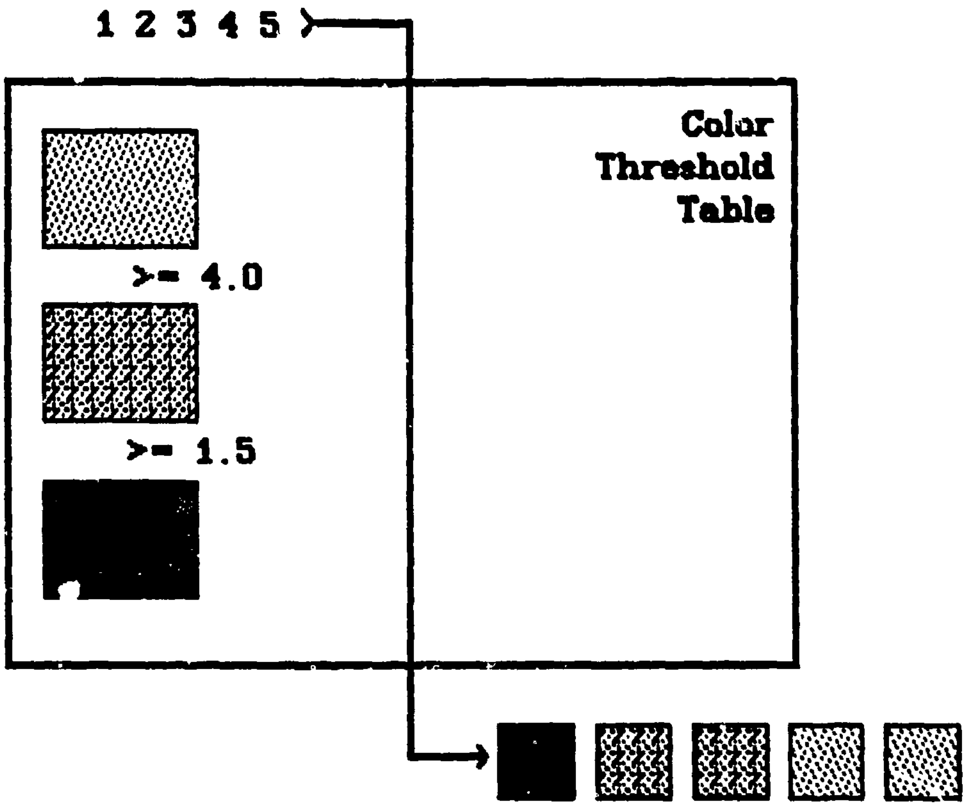


Figure 4. Color threshold table schematic. This table determines the color that will be used to display an object. Depending on the input data value, the output color will be different. This shows the output for an example data stream.

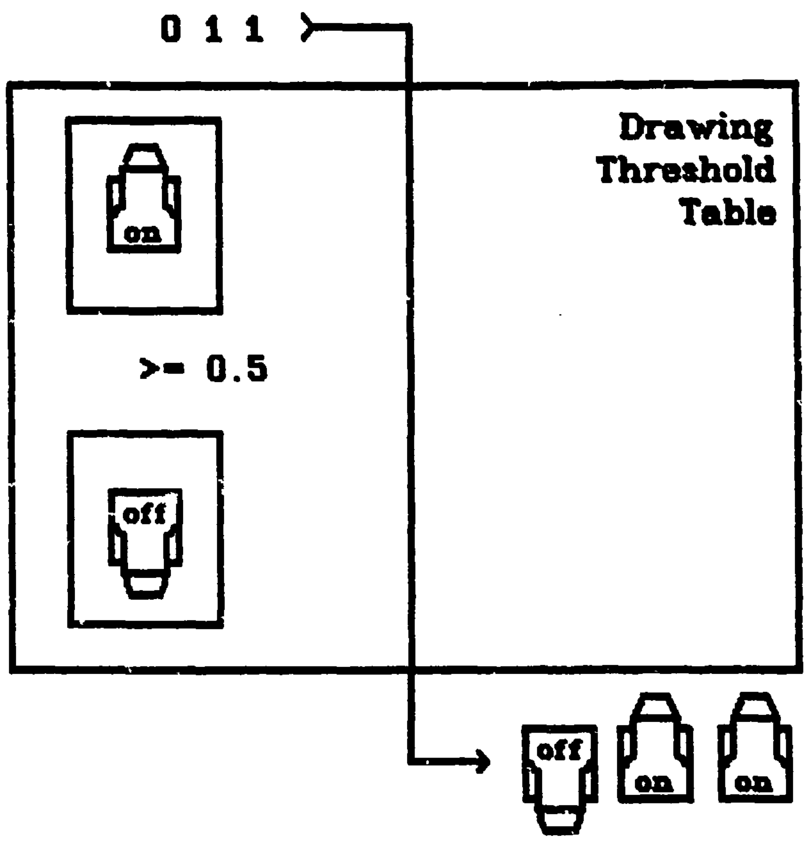
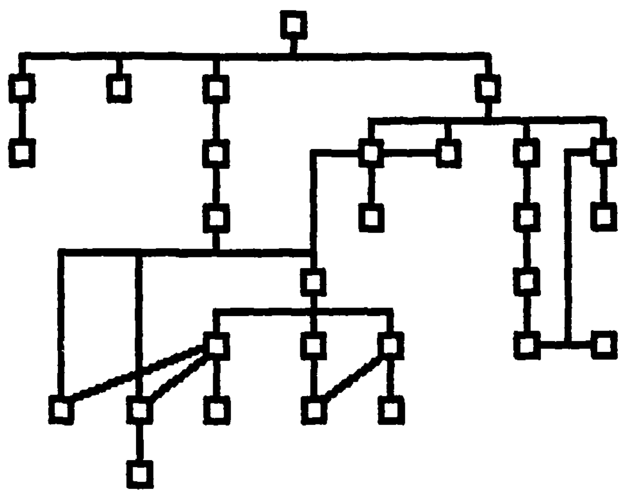
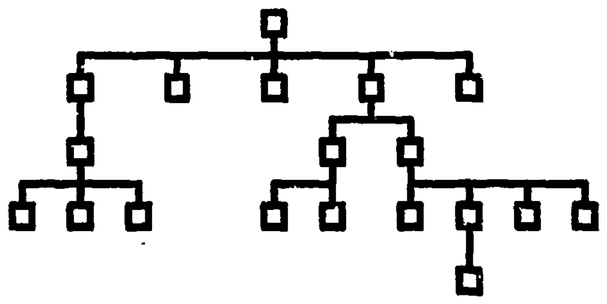


Figure 5. Drawing Threshold Table. The table determines the drawing (icon) that will be used to display a datum value. This shows the output for an example data stream.



(a)



(b)

Figure 6. Trees showing menu structure for DV-Graph for Phase II(a), and for Phase III(b). In the menus represented in (a), typical interactions involve traversing the full depth of the tree. In (b) most interactions only involve the first two levels. Furthermore, in (b) every second level node is immediately accessible from any node in the tree.

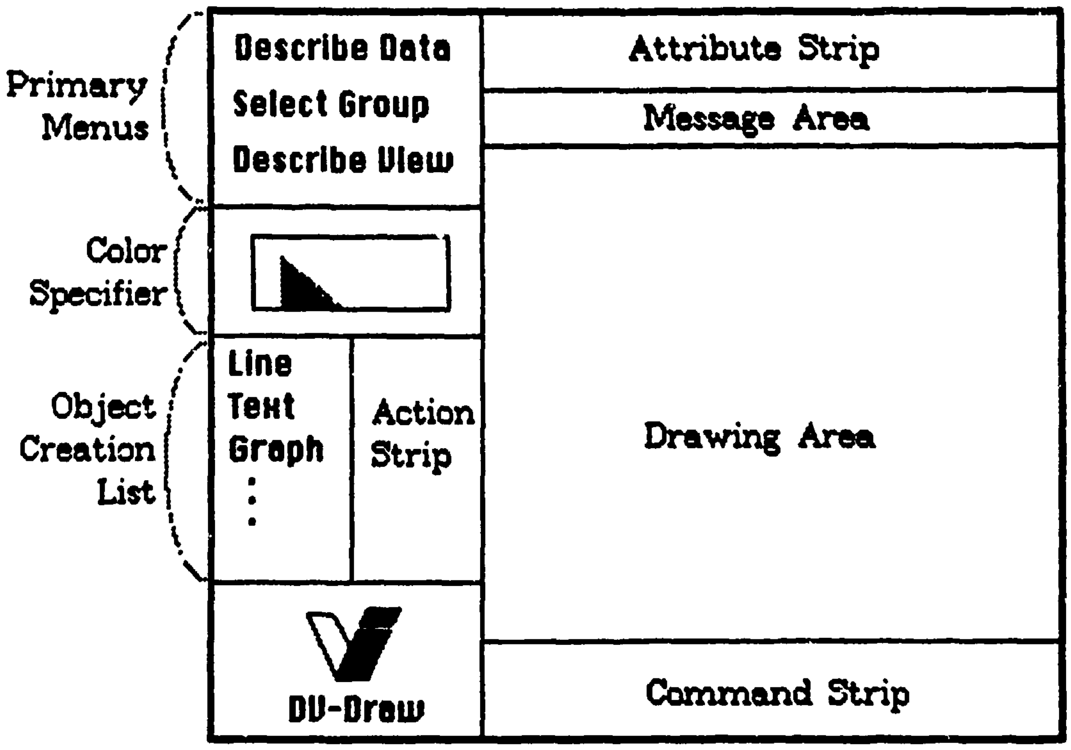


Figure 7. Layout of DV-Draw menu

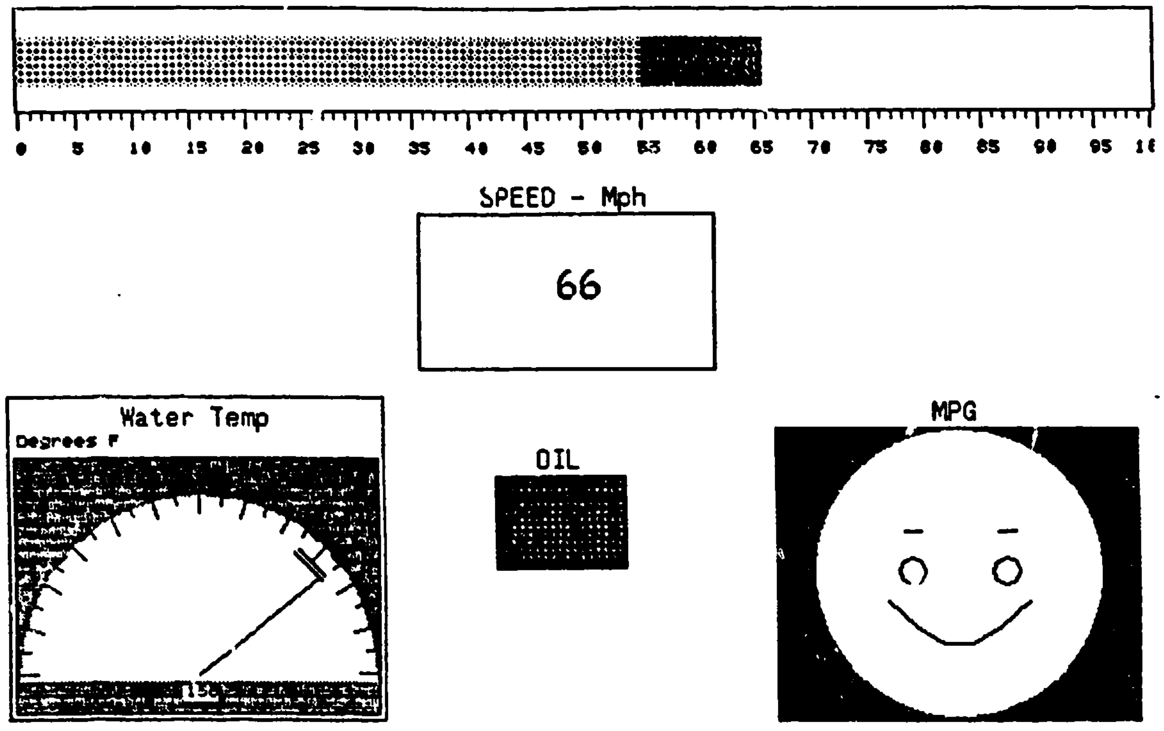


Figure 8. Test display for experiment 1. After undergoing a self-paced tutorial, subjects were asked to create this display using the DataViews editor. This displays the data that you would expect to see on a car dashboard. The speed is represented by a bar graph and a digital display; the water temperature is represented by a meter with history; the oil pressure is represented by a rectangle that changes color according to the datum value; and the gas consumption is represented by a face, where a smiling face corresponded to good gas consumption and an angry face corresponded to bad gas consumption. This display is simple enough that an expert user of DataViews could create it in 10 to 15 minutes.

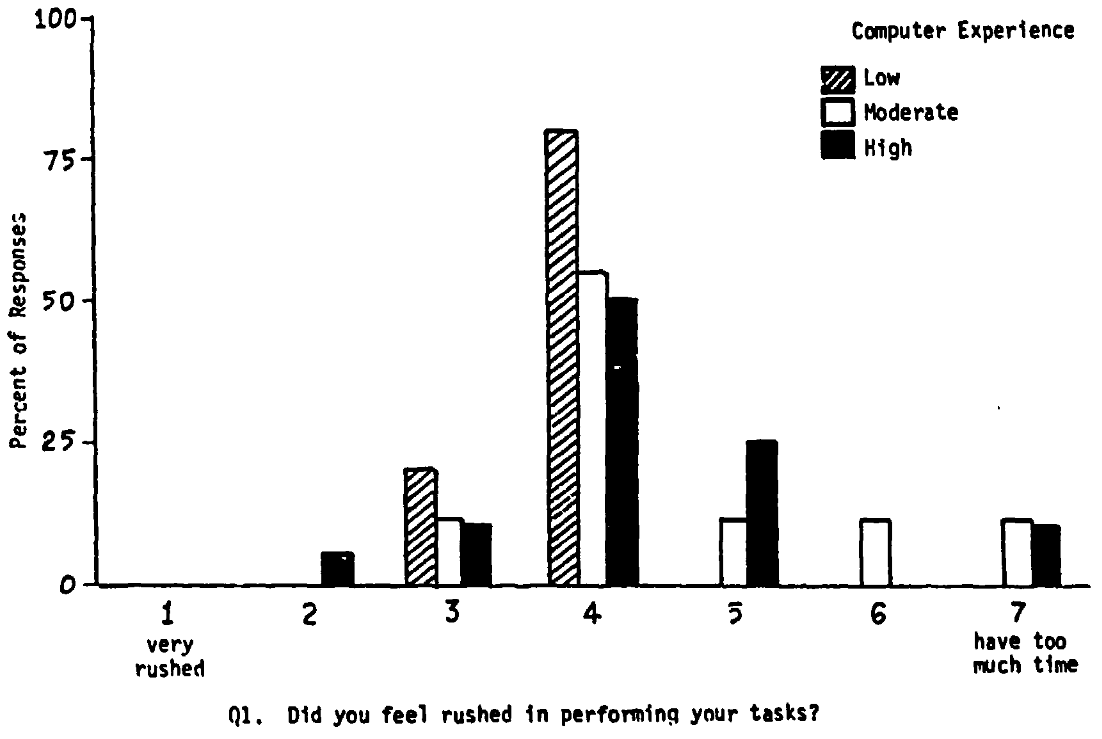


Figure 9. Response to question 1 of questionnaire for experiment 1.

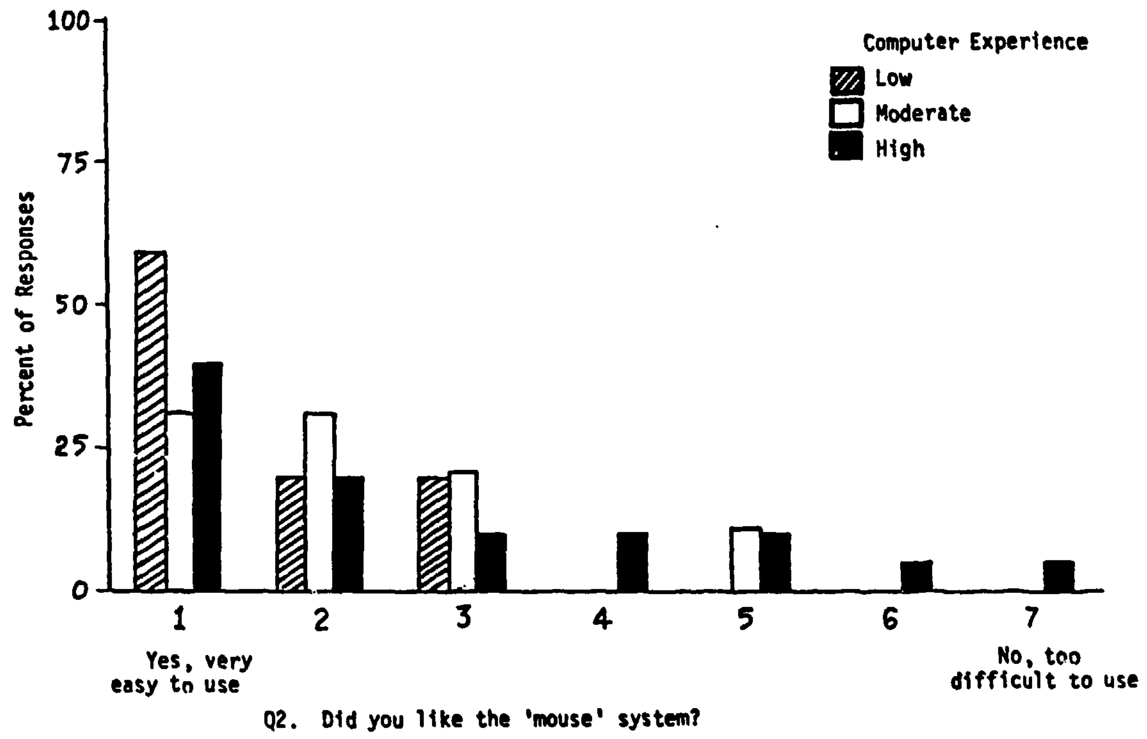


Figure 10. Response to question 2 of questionnaire for experiment 1.

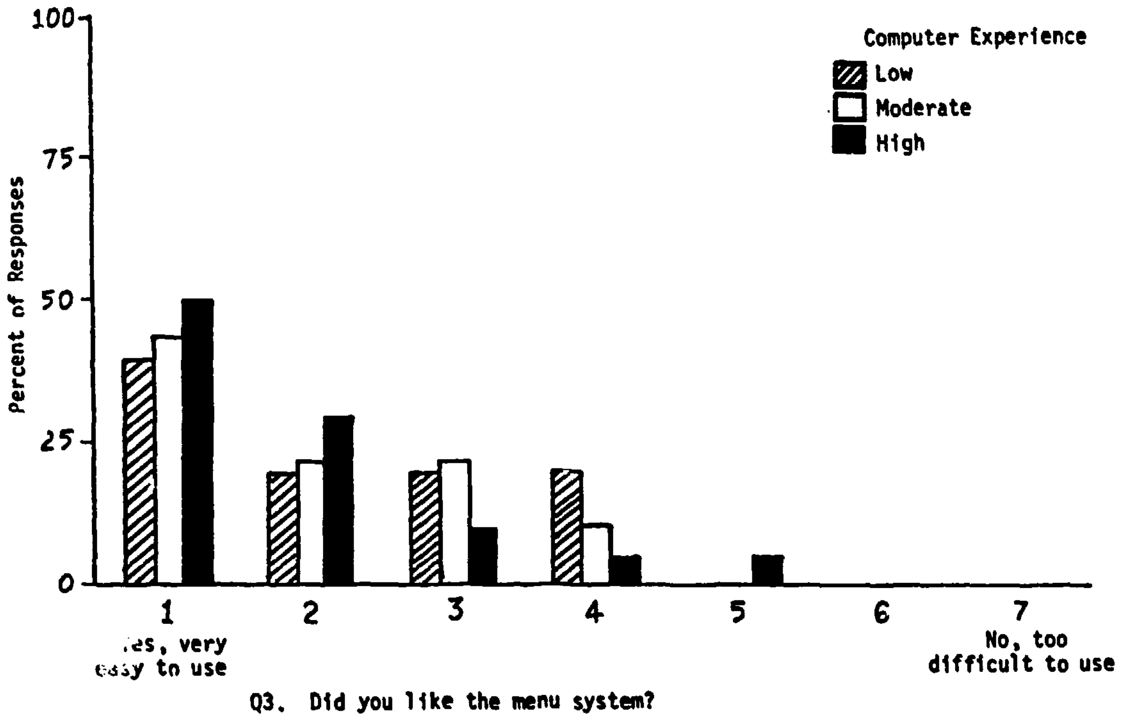


Figure 11. Response to question 3 of questionnaire for experiment 1.

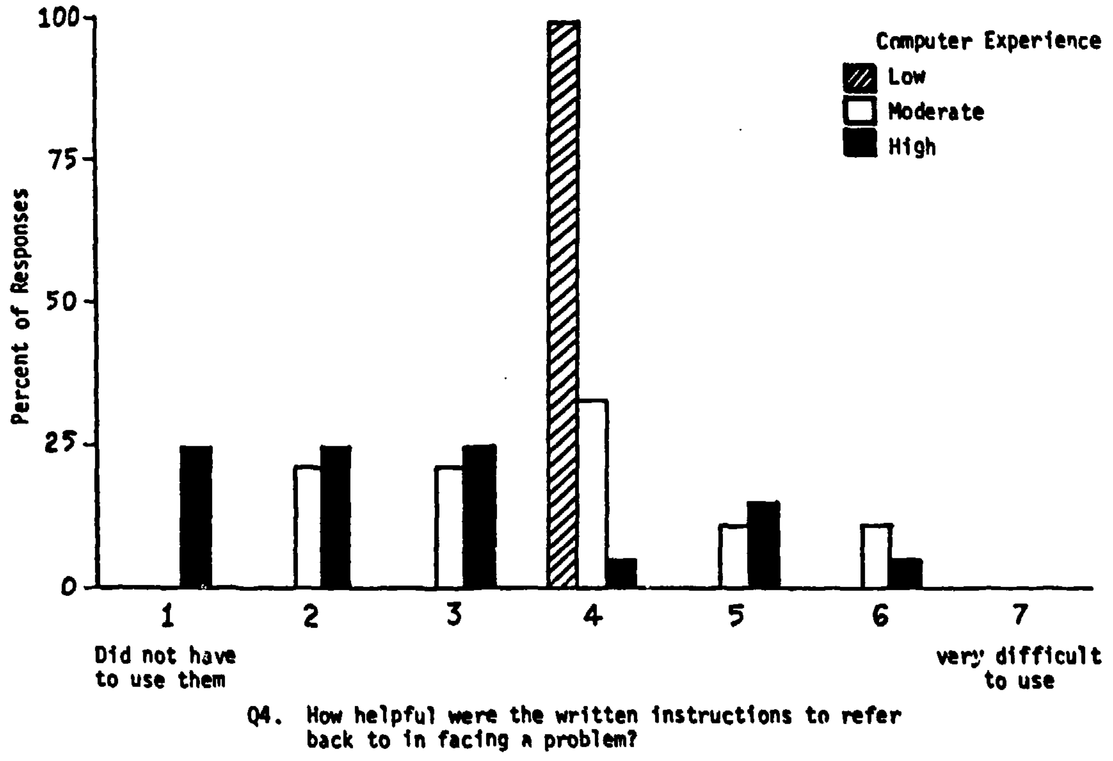


Figure 12. Response to question 4 of questionnaire for experiment 1.

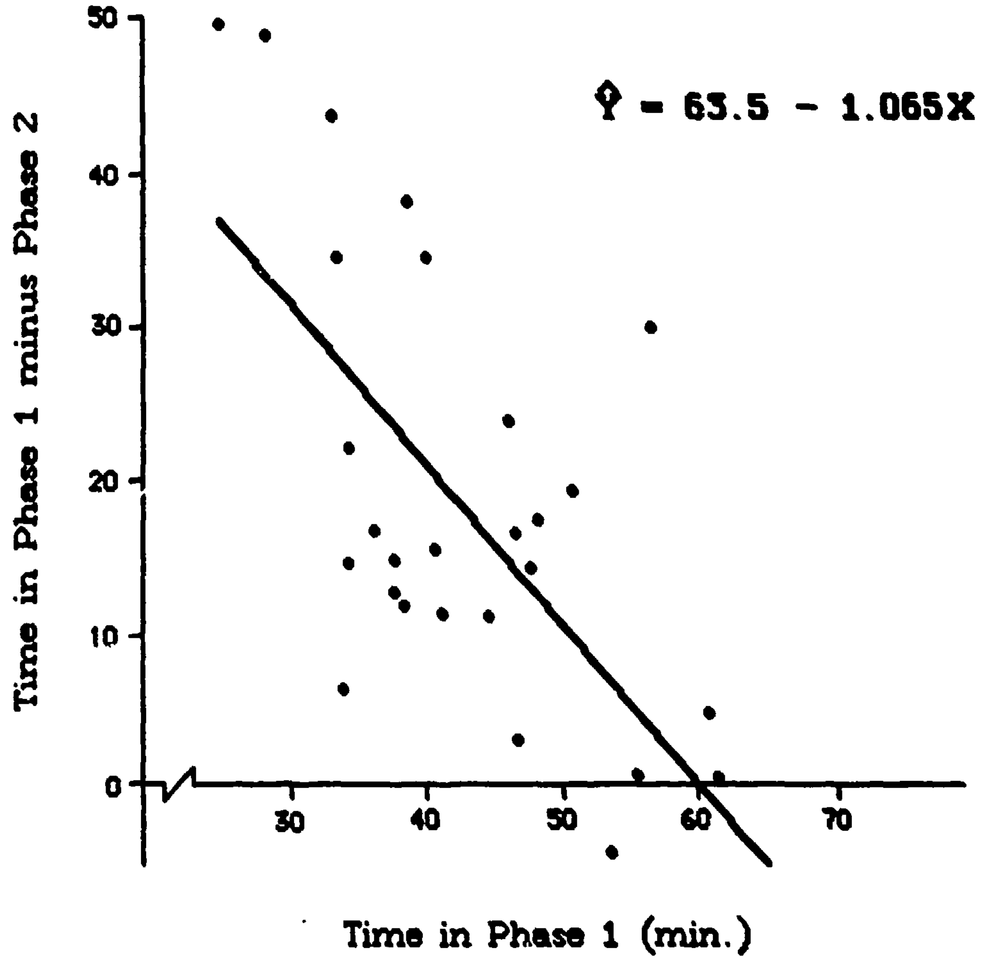


Figure 13. Relationship of time to learn the system (Phase 1), with the additional time required to create the test display (Phase 2). This shows that subjects who took more time to learn the system, took less time to create the test display.

No Colors	Binary Color	Several Colors
ADP	DPA	PAD
APD	DAP	PDA
DPA	PAD	ADP
DAP	PDA	APD
PAD	ADP	DPA
PDA	APD	DAP

A = Analog Display
D = Digital Display
P = Pictorial Display

Figure 14. Latin Square Design for second experiment. This shows the order of presentation of the three displays for each subject. The forty-eight subjects were divided into three groups of eighteen. One had no redundant color encoding, the second had redundant binary color coding, and the third had several redundant colors. Each group was divided into six subgroups of three. Each subgroup worked with the three displays in a different order, as shown above.

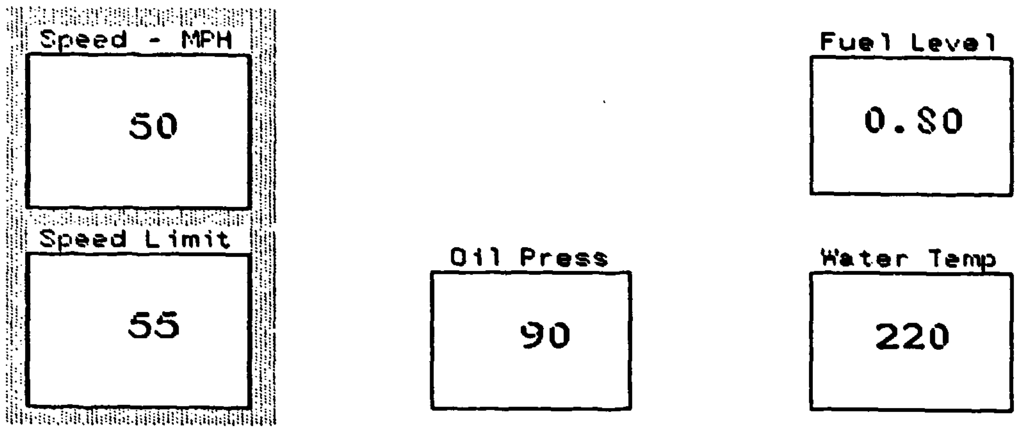


Figure 15. Digital Data Display for experiment. The subject is to monitor the data, responding when the speed exceeds the speed limit and when the oil pressure, fuel level, or water temperature falls outside the acceptable range.

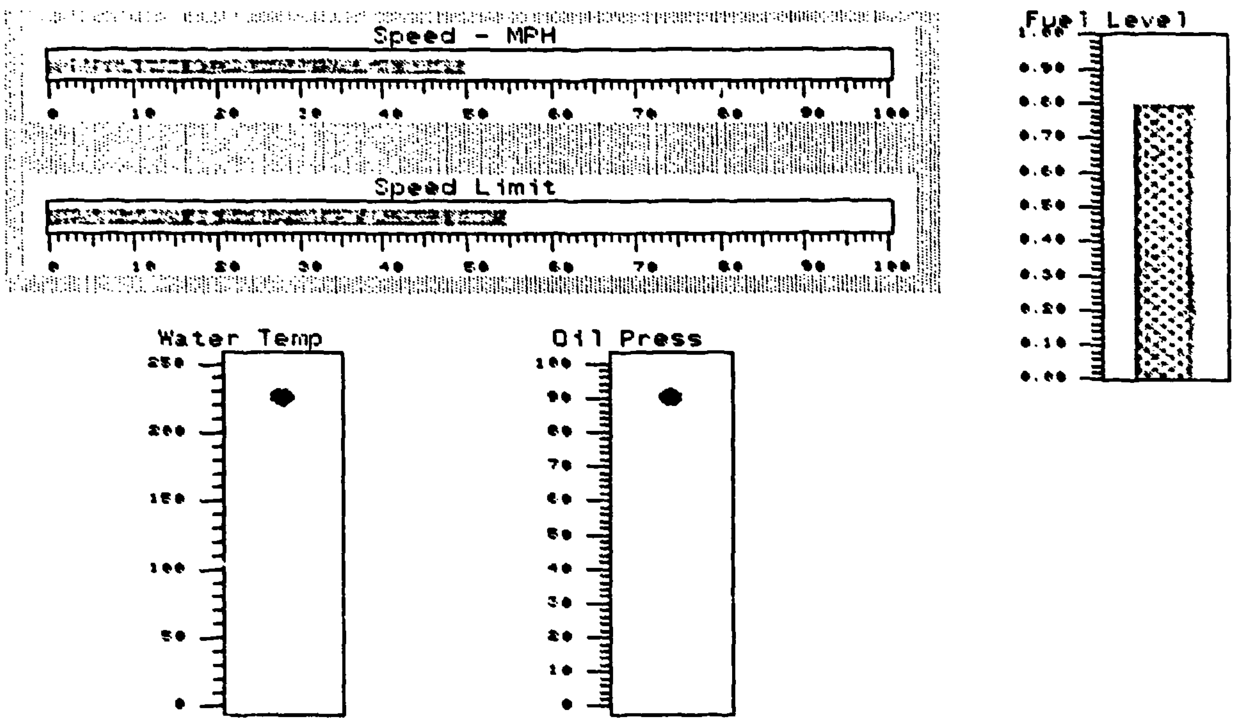


Figure 16. Analog Data Display of the values displayed in Figure 15. The bars change length as the values change, and the points move vertically to indicate the water temperature and oil pressure.

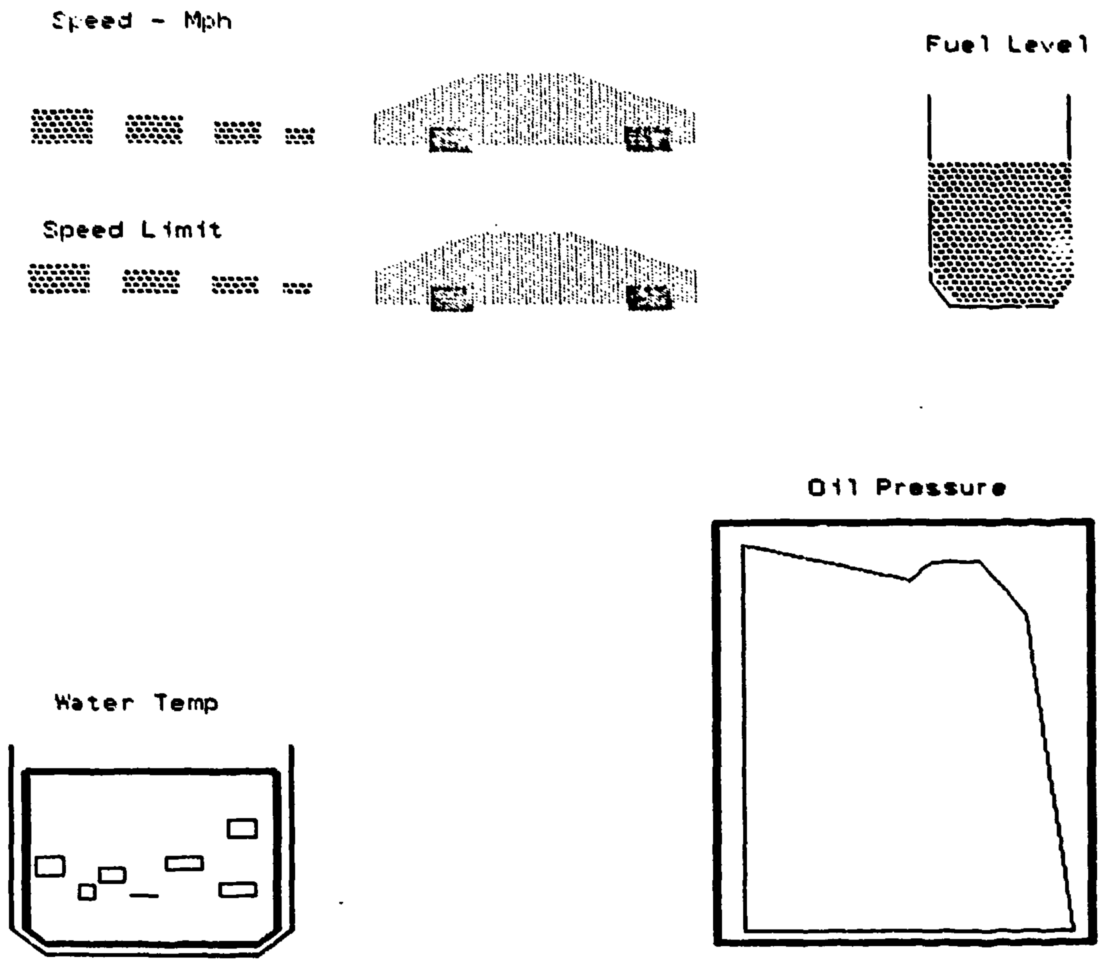


Figure 17. Pictorial display of the values displayed in Figure 15. Speed is indicated by the number of rectangles behind the car icons. Fuel level is indicated by the amount of fluid in the container. Water temperature is indicated by the number of "bubbles" in the container. Oil Pressure is indicated by size of the polygon in the box; as pressure increases the polygon expands to almost fill the box. These displays "discretized" the data, in that the pictures did not change continuously; that is, the pictures were implemented as drawing threshold tables. This means that the pictures only changed the data crossed certain thresholds. There were eight thresholds for each display.

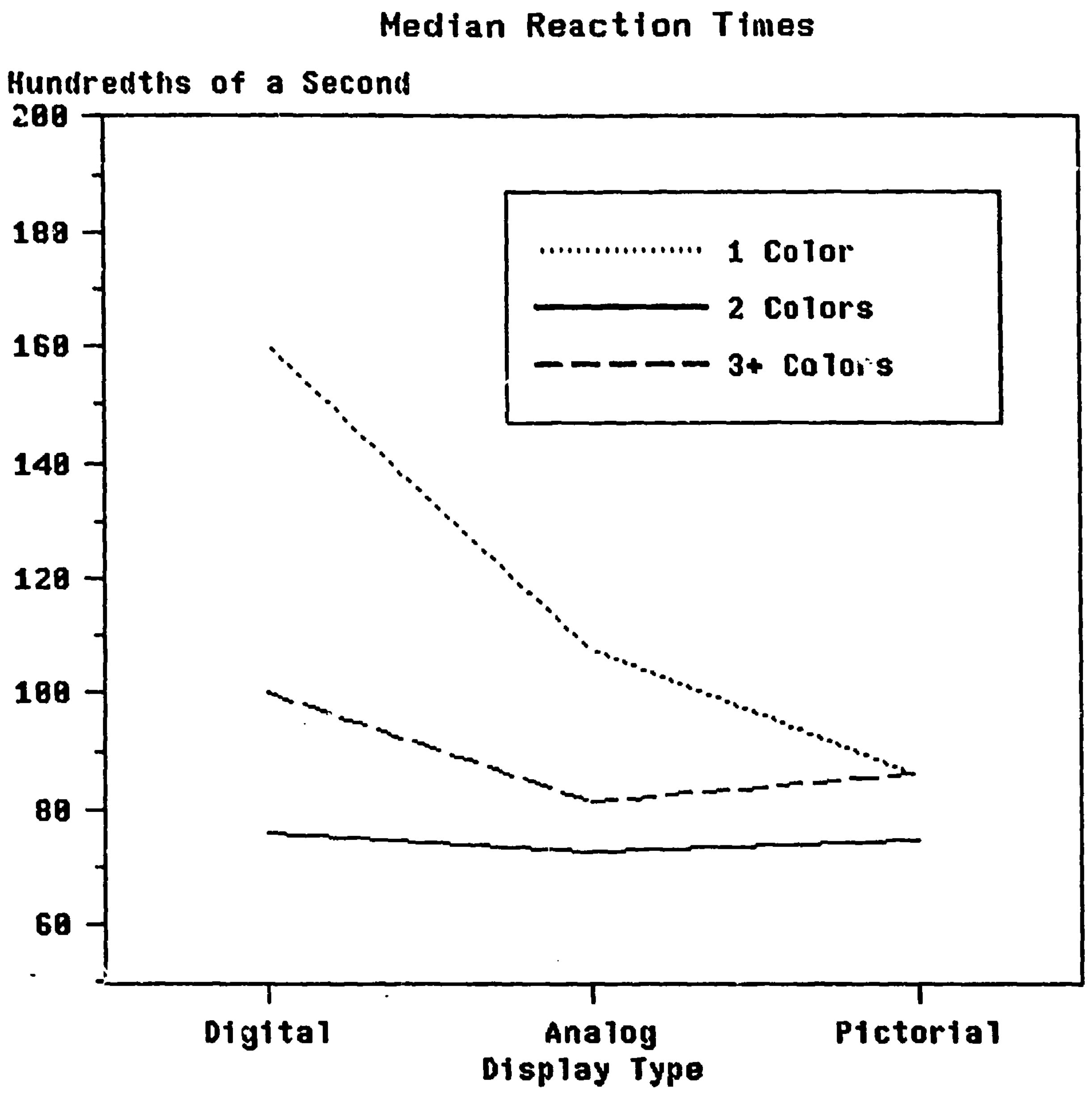


Figure 18. The ef ct of color coding and display type on reaction time. The reaction time for a subject is the median of all the reaction times for the subject for the given display condition. (This is to minimize the effects of misses and temporary lapses of attention.) Then the mean is taken of all the subjects in the group, to derive the reaction time for the display condition. This graph ignores presentation order effects.

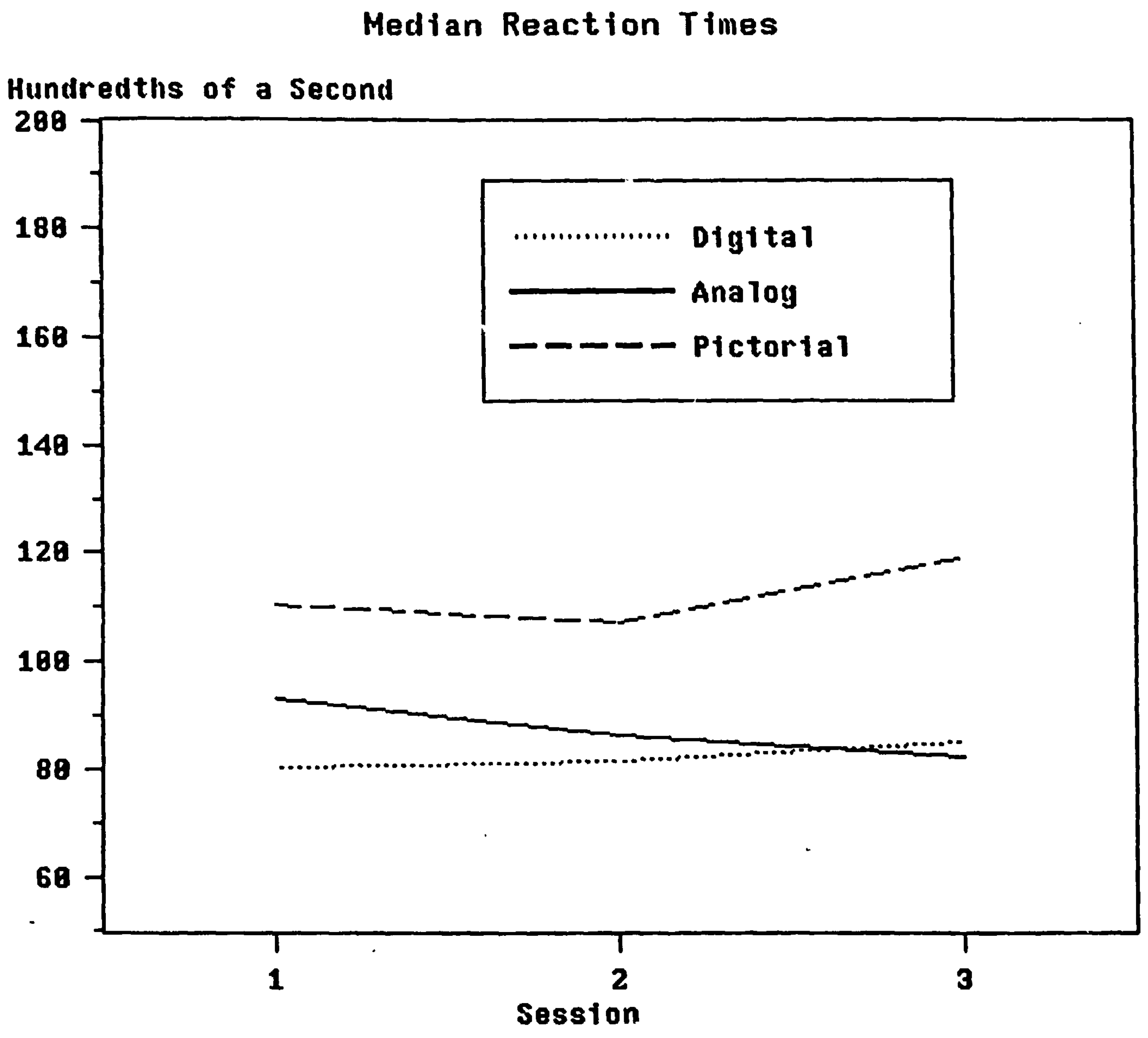


Figure 19. The effect of presentation order and display type on reaction time. Reaction time is calculated as described for Figure 18. This graph ignores the color coding effects.

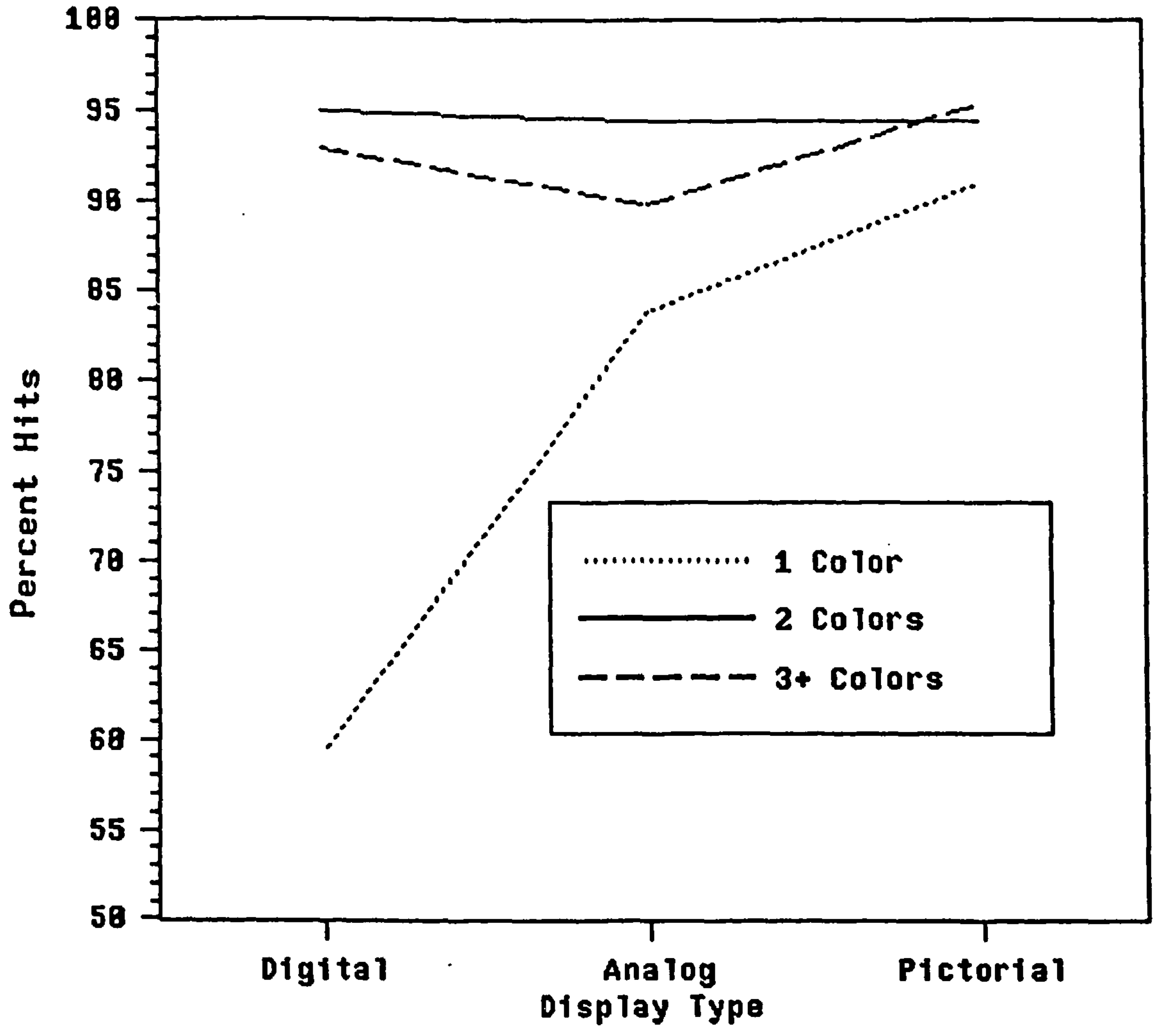


Figure 20. The effect of color coding and display type on accuracy, where accuracy is represented by the percentage if significant events detected by the subject.

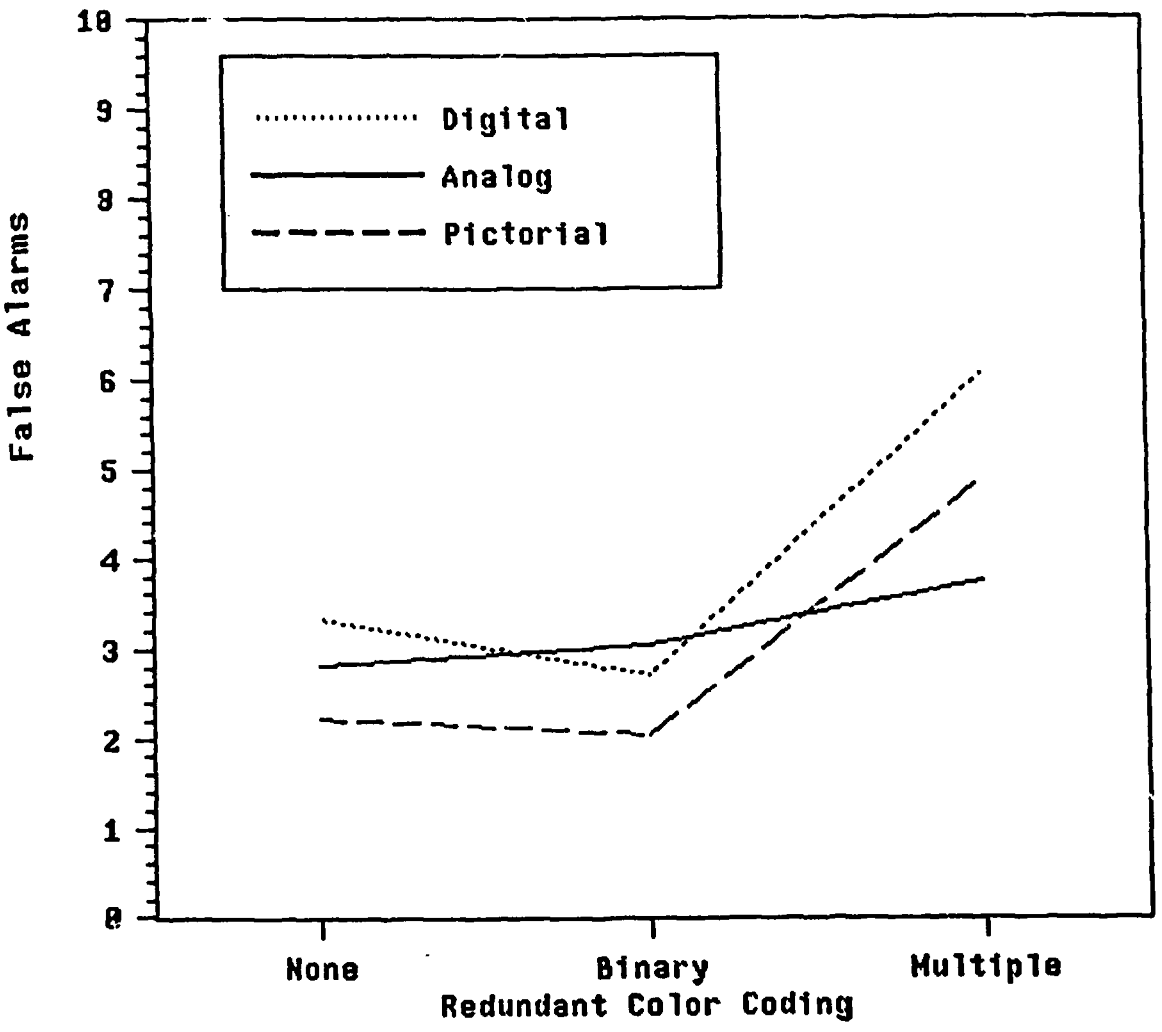


Figure 21. The effect of color coding and display type on the number of false alarms, where subjects reacted to non-significant changes in data.

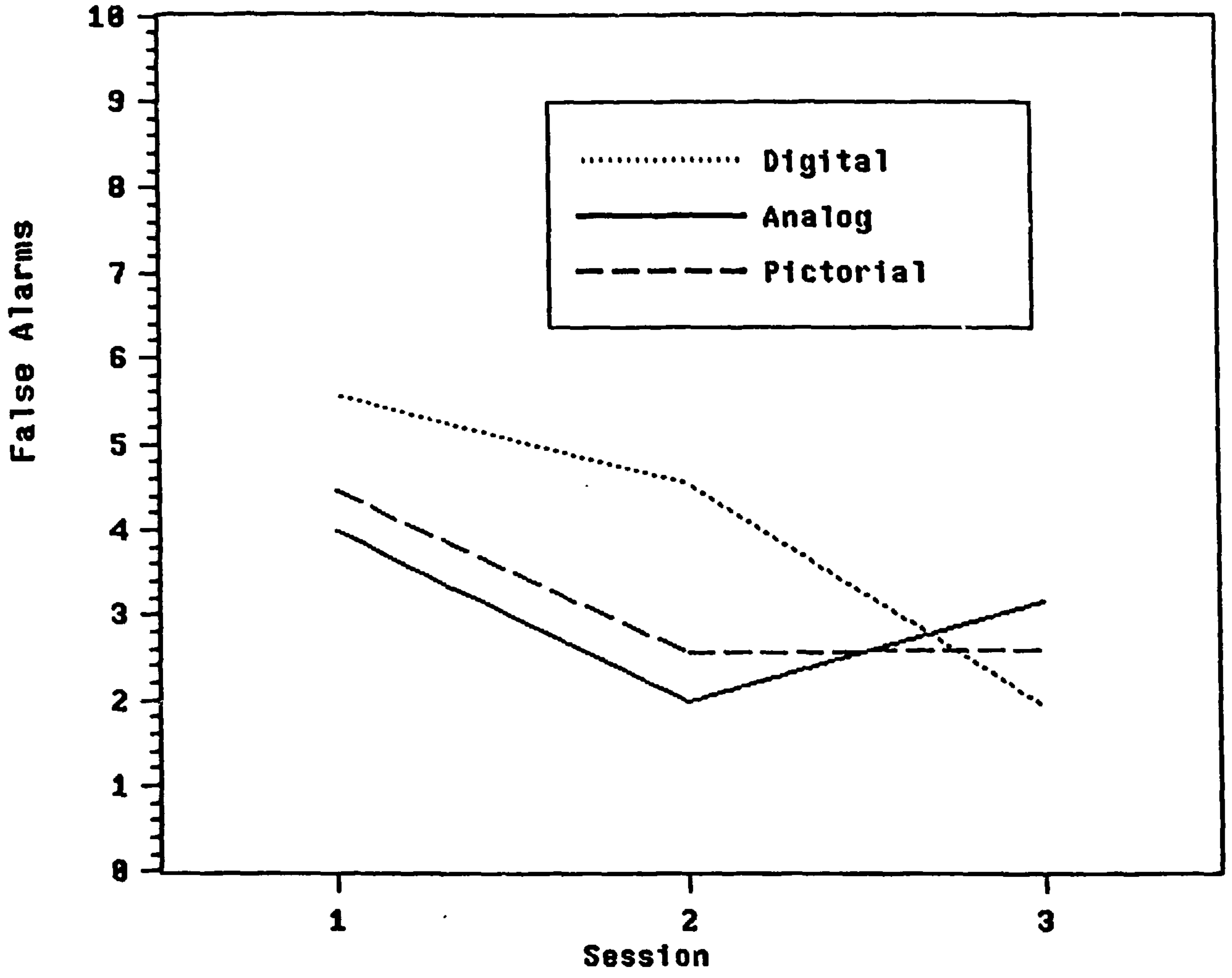


Figure 22. The effect of presentation order and display type on the number of false alarms.

Data Obtained from the Questionnaire

	Overall		No Color Cues		Binary Color Cues		Color Continuum	
	N	%	N	%	N	%	N	%
AGE								
Mean	26.48		26.39		26.22		26.83	
S.D.	5.08		3.36		5.35		6.35	
SEX								
Male	31	57	10	56	12	67	9	50
Female	23	43	8	44	6	33	9	50
HANDEDNESS								
Right	43	81	16	89	15	83	12	71
Left	8	15	1	6	2	11	5	29
Ambidex.	2	4	1	6	1	6	0	0
EASIEST FORMAT TO USE								
Analog	16	30	5	28	5	28	6	33
Digital	12	22	0	0	5	28	7	39
Pictorial	26	48	13	72	8	44	5	28
FORMAT LIKED THE BEST								
Analog	13	30	6	33	3	17	4	22
Digital	11	22	0	0	5	28	6	33
Pictorial	30	48	12	67	10	56	8	44
DIFFICULTY OF TASK								
Extremely Easy								
1	4	8	2	11	1	6	1	6
2	8	15	0	0	4	22	4	24
3	11	21	2	11	5	28	4	24
4	18	34	8	44	4	22	6	35
5	10	19	5	28	3	17	2	12
6	2	4	1	6	1	6	0	0
7	0	0	0	0	0	0	0	0
Extremely Difficult								

Table 1. Responses to the questionnaire for experiment 2. (This questionnaire is contained in Appendix D.)

Appendix A
Menu Language Elements and Syntax

The Menu Specification Language is broken down into the following elements. Listed in logical order they are:

Viewports
Viewport Attributes
Viewport Lists
Strings
Global Variables
Menus
Key Bindings
Action

The syntax for combining these elements are:

Formal Arguments
Local Variables
Call Specifications
Actual Arguments
Item Specifications

Below we describe these objects in greater detail and give examples of their use.

Elements

Viewport

```
VP:<name> <x0> <y0> <x1> <y1> ;
```

X0,y0,x1,y1 are integers 0-32767 or floats 0.0-1.0. X0 and y0 correspond to the lower left corner while x1 and y1 correspond to the upper right corner of the viewport. The viewports are specified in device-independent form. The viewports are often relative to other viewports as well. The lower left corner is at 0,0 and the upper right at 1.0,1.0.

Viewport Attributes

```
VPA:<name> [ FORE: <red> <green> <blue> ]
    [ BACK: <red> <green> <blue> ]
    [ CHAR: <x-size> <y-size> <font> ]
    ;
```

The **VPA** describe how information is displayed in a viewport. They describe:

The colors used. Red, green, and blue are intensities in the range 0 to 255. **FORE** is the foreground color. **BACK** is the background color.

The text style used. **CHAR** specifies the size of the text and the font.

Viewport List

VPLIST:<name> <viewport spec list> ;

The <viewport spec> in the <viewport spec list> is of the form

[**VPA**:<name>] **VP**:<name>

A viewport list is generally used to define viewport pools for menus. The viewport attributes are inherited from left to right. If none is specified the system uses **VPA**:defaultitem. The system provides a default value for **VPA**:defaultitem but the interface designer may override this default by simply defining this viewport attribute explicitly.

String

STR:<name> "<some-string>";

Some-string may contain any characters except quotes (i.e., "). String constants are useful when long strings are used in several places.

Global Variables

GLOBAL: (<variable declaration>,...,<variable declaration>)

or

EXTERNAL: (<external declaration>,...,<external declaration>)

In the first case, a variable is created at run time of <size> entities of type <type>. The types are **CHAR, SHORT, INT, LONG, FLOAT,** and **DOUBLE**. The entities are initialized to values <v0> to <vn>. If no initial values are specified all cells are initialized to 0. If too few initial values are specified all remaining cells are initialized to the last initial value. This permits the initialization of an array to a non-zero constant. The <vi> elements may be integers, floats, or strings (in which case the cell is set to the address of the string).

In the second case EXTERNAL:<name> becomes an alias for addressing the external variable <externalname>.
When <externalname> is left out, then <name> is the name of the external as well. The type is the same as above. The advantage to using this alias is that the variable can be indexed to refer to individual elements of array

variables and the size of each element will be known from the type (which is not available for externals otherwise).

Menus

```
MENU:<name> ( <formal arg list> ) <optional local variable>
 {
 AT: <viewport spec> ;
 [ INITACT: <call spec> (<arg list>); ]
 [ TERMACT: <call spec> (<arg list>); ]
 [ PREKEYS: KBND:<name> (<arg list>); ]
 [ ITEMS: ( <item spec list> ) ]
 [ POSTKEYS: KBND:<name> (<arg list>); ]
 [ DEFAULTACT: <call spec> (<arg list>); ]
 [ VPPOOL: VPLIST:<name>; ]
 }
```

The <formal arg list> has formal arguments <arg1> to <argn> and can define local variables which exist only while the menu is active.

Any *'s in the local variables declaration are currently ignored but are useful for remembering what level of indirection an argument is supposed to have.

The **AT** field is the only required field and specifies the viewport in which the menu is to be displayed.

The **INITACT** field is an initial call which is executed in the context of the menu before the menu is displayed. This can be used for fancy overriding of display functions or to compute initial values of temporary variables.

TERMACT similarly is called when the menu is about to be exited. This call can be used to clean up or close files after a quit.

PREKEYS specifies the highest precedence key bindings; **KBND**:<name> is the set of key bindings and <arg1> to <argn> are actual arguments sent to the keybindings. **POSTKEYS** is identical except that these key bindings have lowest precedence. The item buttons have precedence between these.

ITEMS specifies the logical choices available in the menu and what to do if the choice is selected as well as some help for the choice. This category is used to specify message areas as well as buttons for user selection.

DEFAULTACT is used to declare a call which is executed only if the user has made a selection (via mouse or keyboard) but that selection has not been processed by any of the key or button detection methods.

The **VPPOOL** is a list of viewports which may be allocated to the display of the items. These are the viewports used in multi-page menus. When all vps in the pool are used the page is full and additional items are not displayed until a next page command is executed. Some items, especially messages, will not use vps in this pool, but will have a single **VP** specification in the **AT** field of the <item spec>.

NOTE: Device for menus is now specified only in the global variable M_menudevice. It may be dynamically altered to display menus on different devices at run-time.

Key Binding Specification

```
KBND:<name> (<formal arg list>)
   <key binding spec> <call spec> (<arg list>), ...
   <key binding spec> <call spec> (<arg list>);
```

The keyboard binding specification allows any of the keyboard keys to be bound to any desired call. The <key binding spec> is either a decimal number, a hex number, or single keyboard key enclosed in single quotes.

Actions

```
ACT:<name> (<formal arg list>)
  <optional variable declaractions>
  {
  <act-body>
  }
```

The <formal arg list> specifies formal arguments <arg1> ... <argn>. The optional variable declaractions define local variables which exist for the duration of the action and are initialized when the action is invoked.

The body of the action is essentially a sequence of action elements:

<act-body> :== <act-elem> ... <act-elem>

Each <act-elem> is one of the following:

<call-spec>(<arg1> ... <argn>);
To call a subroutine, action vector or menu;*/

IF(<var>) <act-block> [**ELSE** <act-block>] **ENDIF**
To provide conditional execution;

WHILE (<var>) <act-block> **ENDWHILE**
A for-while loop;

REPEAT <act-block> **UNTIL** (<var>)
For repeat loop;

ENDACT
To return from action or looping in a menu;

QUIT
To return from action vector and, if invoked by a menu,return to the calling menu;

PASSPICK
To return the current user selection for handling by the calling menu.

The <act-block> used with IF, WHILE, and REPEAT is the same as an <act-body> except it can be placed in curly braces ('{' '}') and can thus be empty:

```
<act-block> :== <act-body> | { [ <act-body> ] }
```

Syntax

Formal Arguments

Formal arguments (i.e., <formal arg list>) are defined as:

```
<type> [*'s]<name>
```

The *'s are currently ignored by the computer but are useful to help you remember what level of indirection an argument is supposed to have.

Local Variables

Local variables are discussed under Menu and Action.

Call Specifications

A <call spec> is a call to a routine followed by actual arguments and is written in one of the following ways:

ACT:<name>
MENU:<name>
POLLMENU:<name>
PICKMENU:<name> or **DRAWMENU**:<name>
ERASEMENU:<name> or <user-subroutine>

When the call is invoked, the call-routine is called with the actual arguments. Normally, menus are called with the syntax MENU:<name> and then they are drawn, run, and erased on exit.

Menus to be run concurrently with others are called using the call-syntax POLLMENU:<name>. Such menus must be explictly displayed using the call-syntax DRAWMENU:<name> prior to calling them and must be explicitly erased using the call-syntax ERASMENU:<name> after the last POLLMENU call has been made.

When the initial pick is already known for a menu before it is called, then

PICKMENU:<name> is used to indicate this.

Actual Arguments

Actual arguments can be specified following any <call spec> or **KBND**:<name> reference. The actual arguments can be:

Direct References:

Explicitly defined language entities such as **VP**, **STR**, or **GLOBAL**, etc;
Formal arguments to the entity in which the actual argument is used;
Local variables defined in the entity in which the actual argument is used;

NOTE on calling external subroutines: All direct reference actual arguments are passed by value, i.e., the value, not the address, is put on the call stack, **unless** the entity is a global or local variable which is declared as an array or a **STR**, in which case the address is used. Floats are converted to **DOUBLE** and take up two call stack positions (The C language expects this).

Indexed References:

<var>[n], where <var> is a local variable or a formal argument declared as an array, a **STR**, a **VP**, a **GLOBAL**, or an **EXTERNAL**. [n] references the nth element of the named structure.

NOTE: Element size is as declared for global vars, local vars and formal arguments, **CHAR** for strings, **SHORT** for viewports.

Indirect References:

* followed by any direct reference, (see above) of type **LONG**, taking the contents of the first element of the entity as an address and referencing the **contents** of the addressed location.

NOTE: We assume the addressed location contains a longword. This is due to weakness in the declaration syntax.

Indexed Indirect References:

*<var>[n] meaning the contents of the location addressed by the nth element of <var>; but <var> must be a LONG. (The same warning applies as for Indirect reference.

NOTE: There is no indirect-indexed referencing, i.e. taking the nth element starting at the location pointed to by the contents of the named entity.

Address References:

& followed by any direct reference, (see above) referencing the address of the entity.

Indexed Address References:

&<var>[n] referencing the address of the nth element.

External References:

Subroutine names from the underlying program such as Mdspmsg reference the address of the subroutine or variable.

Menu Item Reference:

ID:<name> to pass the item index of the named item.

Item Specifications

An <item spec> specifies the logical choice for the display box:

```
(
[ ID: <name> ; ]
AT: <viewport spec>; | VPPOOL ;
DISPLAY:  "...";  | STR:<name>; | <INT-fun>(<arg list>);
[ HELP:   "...";  | STR:<name>; | <void-fun>(<arg list>);]
[ ACTION: <INT-fun> ( <arg list>); ]
[ HIGHLIGHT: <void-fun> ( <arg list>); ]
[ PICKLIGHT: <void-fun> ( <arg list>); ]
[ WASPICKED: <INT-fun> ( <arg list>); ]
)
```

The **ID** field is used to name the item. This name can be used in actual argument lists to pass the item index to calls in order to refresh particular messages or buttons.

The **AT** field specifies the viewports in which to display the item. When the **AT** field is set to **VPPOOL** then the actual viewport used is taken from the **VPPOOL**. This is useful with multipage menus and when items are a dynamically varying set.

The **DISPLAY** field is typically a string but may be a function used to display the item. If a dynamically generated list of items is to be displayed, the data structure would be an argument to a special display routine. The **HELP** field specifies how a help item is to be displayed; typically this is also just a string. The **ACTION** field specifies what to do when the item is picked. If the item is intended to be a message area the **ACTION** would, of course, be **NULL**.

The functional arguments are for more complex displays. The display function should use **M_subitemindex** to determine which sub-item to display. If the sub-item index is 0, the function must return the total number of subitems. The function may use **M_subitemindex** directly or have it passed as one of its arguments. The default function will display a single item which is a string pointer. The arguments typically include the item and sub item index, the current item pointer, the current display viewport, attributes of the viewport, the current cursor location and the character associated with the last user pick. These globals are provided by the system in the globals **M_itemindex, M_subitemindex, VM_itemvp, VM_itemvpa, VM_cursorpoint,** and **M_cursorchar**.

The **HIGHLIGHT** function is used to indicate which item would be picked if the user made a selection at that point, and the **PICKLIGHT** is used to indicate the item has been selected. The help function is used to display help text.

WASPICKED returns an integer in the range **MNOTPICK** to **MSUREPICK** indicating the importance of or confidence in the pick of the corresponding item. By default **WASPICKED** returns **MNOTPICKED** when the user's selection falls outside the viewport of the current item and **MDEFAULTPICK** (it lies between **MNOTPICKED** and **VMSUREPICKED**) if it is inside the item viewport. The system has access to the current cursor location through the variable **M_cursorpoint**.

Functions for building and using a doubly linked list of items will be provided for dynamic item list management.

Sample MSL Menu

Below is an simple menu to show how the MSL can be used. Some of the code may be inefficient because its main purpose is to show language elements in use. The comments embedded in the code will help you find examples of the elements that are included.

Here is a picture of the sample menu that results from the code:

There is one menu with four items and a keybinding (for quiting). When items in the menu are picked, the old value of a variable is printed out, the name of the item picked is printed and the variable is given a new value.

```
/* define the relevant viewports */
/* a viewport in upper right*/
VP:main   0.51 0.51 1.00 1.00 ;

/* four item viewports */
VP:item1  0.10 0.40 0.40 0.60 ;
VP:item2  0.10 0.10 0.40 0.30 ;
VP:item3  0.60 0.40 0.90 0.60 ;
VP:item4  0.60 0.10 0.90 0.30 ;
```

```
/* a title viewport for the menu */
VP:title   0.10  0.80  0.90  1.00 ;

/* a viewport to hold the start point for output text */
VP:output  100 100 100 100;

/* define viewport attributes needed */
VPA:menus  CHAR: 1 1 1  FORE: 220 200 0  BACK: 200 0 0 ;
VPA:titles  CHAR: 1 1 1  FORE: 10 10 10  BACK: 150 0 0 ;
VPA:items  CHAR: 1 1 1  FORE: 0 0 0  BACK: 130 110 0 ;

/* define any string constants needed */
STR:maintitle "Main Menu";

/* define the keyboard bindings needed */
/* "arg" passed in when called */
KBND:quitkeys(LONG arg)
   03 ACT:quit(arg),  /* right mouse button to quit*/
   'q' ACT:quit(arg);  /* 'q' to quit */

/* Define global variables */
GLOBAL:(
  LONG global_var (0), /* long integer */
  CHAR global_str [10](0) /* 10 element char array */ );

/* define the menu */
MENU:main()
  LONG local_var (85); /* Initialized to 85 */
  {
  AT: VPA:menus VP:main;
  PREKEYS: KBND:quitkeys( &local_var);
  INITACT: ACT:maininit ( local_var ) ;
  TERMACT: ACT:mainterm ( local_var );

  ITEMS:
    ( ( AT: VPA:titles VP:title; DISPLAY: STR:maintitle;
      /* no action, display-only item */ )

      ( AT: VPA:items VP:item1; DISPLAY: "dummy item 1";
        ACTION: ACT:item1 (&local_var); )

      ( AT: VPA:items VP:item2; DISPLAY: "dummy item 2";
        ACTION: ACT:item2 (&local_var); )

      ( AT: VPA:items VP:item3; DISPLAY: "dummy item 3";
        ACTION: ACT:item3 (&local_var); )

      ( AT: VPA:items VP:item4; DISPLAY: "dummy item 4";
        ACTION: ACT:item4 (&local_var); )
    )
   DEFAULTACT:  ACT:dfltact(local_var);  }
```

```
ACT:quit(LONG var_ptr) {
  GRmove(VP:output);
  printf("var=%d. now quit called",*var_ptr);
  Mlongassign(var_ptr,0);
  QUIT
}

ACT:item1(LONG var_ptr) {
  GRmove(VP:output);
  printf("var=%d. now item1 called",*var_ptr);
  Mlongassign(var_ptr,1);
}

ACT:item2(LONG var_ptr) {
  GRmove(VP:output);
  printf("var=%d. now item2 called",*var_ptr);
  Mlongassign(var_ptr,2);
}

ACT:item3(LONG var_ptr) {
  GRmove(VP:output);
  printf("var=%d. now item3 called",*var_ptr);
  Mlongassign(var_ptr,3);
}

ACT:item4(LONG var_ptr) {
  GRmove(VP:output);
  printf("var=%d. now item4 called",*var_ptr);
  Mlongassign(var_ptr,4);
}

ACT:maininit(LONG var) {
  GRmove(VP:output);
  printf("var=%d. menu starting.", var);
}

ACT:mainterm(LONG var) {
  GRmove(VP:output);
  printf("var=%d. menu done.", var);
}

ACT:dfltact(LONG var) {
  GRmove(VP:output);
  printf("var=%d. default action.", var);
}
```

Appendix B
Problems/Solutions

This appendix describes some of the problems that subjects encountered in running DataViews. Many of these comments were addressed in the next release of the product. Below we list the problems and how they were addressed in the new release.

Subjects confused the "Choose Another Graph" command with the "Select Graph Type" command.
The "Choose Another Graph" command was made unnecessary and removed.

Some subjects became confused when changing display viewports.
Display viewport setting was simplified.

Subjects accidentally deleted the wrong display.
The system was changed so that this would be less likely to happen and an "Undelete" command was added.

Subjects missed error messages, because they disappeared as soon as the mouse is moved.
Error messages now require a keypress before they disappear and before the user may continue editing.

Subjects confused pressing <RETURN> after entering text (the proper response) with pressing a mouse key (appropriate in other circumstances), which caused the text to have to be re-entered.
Mouse press during text entry now has no effect.

Subjects could not cancel a text entry menu selection; they had to retype the string.
Improved for some text entry, where entering an empty string cancels the command. This doesn't work for those text fields that may be null. This type of error happens infrequently.

After entering text, but before pressing <RETURN>, some subjects thought they were done and tried to locate the cursor, but it was not visible, because the system was still in text entry mode.
Still a problem for the naive user.

Subjects were confused by an apparent inconsistency between the way color thresholds were added and they way they were changed.
The prompt messages were changed.

When editing color threshold table and trying to modify a color in the table, some subjects were confused by the prompt "pick color to change"; they went to pick from the color palette instead of the

color threshold table. (They were to pick from the palette after they pointed to a color to change in the threshold table.)
Still a problem for naive users.

Subjects were confused by the normalized thresholds of the color threshold tables.
Color threshold tables were changed to have the same range as the variables.

Subjects were confused by labelled text fields, like the graph's title, which has the word "Title:" adjacent to a field containing the actual title. The subjects tried to pick the word "Title:" to change the title, when they should have picked the field containing the actual title.

The system now allows picking of either the text field and the label field.

No reassuring messages appeared during delays after selecting "Run" or "Add a graph" menu options.
These delays have been removed except in one case ("Run" when the data source is a pipe), which is still a problem.

Locations of value and time axes are inconsistent across display types.
This is necessary, because of the different classes of display type. Within display class the menus are identical.

Subjects were confused when defining a viewport as to which corners should be specified.
The system now makes it clear that any opposite corners will do.

Message "Viewport too small" disappears to quickly to be read.
System now waits for a keypress before continuing.

Subjects had difficulty choosing effective colors since they had to "run" the system to see the displays.
System now gives better color feedback, so running the display is unnecessary to see the combined colors.

Subjects occasionally edited wrong variable.
Variable editing mechanism simplified.

Subjects were unable to compare displays without "running"; they could only see one display at a time.
System now gives better feedback. Subjects can see the full display layout without "running," and a special preview command was added.

"Preview the graph" command didn't work properly.
This was fixed.

"Delete color" command for color threshold tables didn't work properly.
This was fixed.

Subjects accidentally deleted displays and they could only recover by recreating the display.
An "Undelete" menu choice was added, appearing where the "Delete" menu choice had been.

Mouse button allowed picking ahead, which led to unpredictable results.
Mouse button no longer allows pick-ahead.

Some subjects found some of the icons confusing.
Icons changed to differentiate them better. (Even with no change, subjects found icons vastly better than text.) Nevertheless, this could still be a problem for the naive user.

Subjects found the viewport layout mechanism difficult to use.
The display viewport layout mechanism was simplified and improved, mostly by making the layout area much larger.

Appendix C
Questionnaire for Experiment 1

Name ______________________________ Date ______________
(Last) (First) (MI)

Age ______ Sex _____ Phone Number ____________

Academic Major (if student) ______________________ College semesters Completed ______

Describe type of previous computer use: ______________________________
(programming, games, word processing)
____

Type of computer owned ______________________________
(or rented)

Ever used a Mouse before? Y_____ N_____

With respects to the study that you have just completed, please answer the following:

1. Did you feel rushed in performing your tasks? (circle one)

   1 ---- 2 ---- 3 ---- 4 ---- 5 ---- 6 ---- 7
   very rushed — have too much time

2. Did you like the 'mouse' system?

   1 ---- 2 ---- 3 ---- 4 ---- 5 ---- 6 ---- 7
   yes, very easy to use — no, too difficult to use

3. Did you like the menu system?

   1 ---- 2 ---- 3 ---- 4 ---- 5 ---- 6 ---- 7
   yes, very easy to use — no, too difficult to use

4. How helpful were the written instructions to refer back to in facing a problem?

   1 ---- 2 ---- 3 ---- 4 ---- 5 ---- 6 ---- 7
   did not have to use them — very difficult to use

5. What instructions did you find to be the most confusing? (Identify them by stating the procedure that they were associated with)

6. Using the pen provided, place a (+) on the picture of the instrument panel by those displays you feel were easy to develop. Put a minus sign (-) by those that were difficult to develop.

Appendix D
Questionnaire for Experiment 2

Subject No ____
Date ________

Name ____
(Last) (First) (MI)

Age ________ Sex ______ Phone Number ________________

Academic Major (if student) ______________________________

College semesters Completed ______________ Handedness: R___ L___ A___

With respects to the study you just completed, please answer the following:

1. Which display format did you find the easiest to use?

   Analog ____ Digital ____ Pictorial ____

2. Which display format did you like the best?

   Analog ____ Digital ____ Pictorial ____

3. How difficult did you find the task?

   1 ---- 2 ---- 3 ---- 4 ---- 5 ----- 6 ---- 7
   extremely easy ... extremely difficult

4. Please rate the difficulty of detecting parameters when the data in a display was out of range in each of the following display formats: (1=easy to detect to 10=hard to detect)

ANALOG	DIGITAL	PICTORIAL
____ Speed	____ Speed	____ Speed
____ Water Temp	____ Water Temp	____ Water Temp
____ Fuel Level	____ Fuel Level	____ Fuel Level
____ Oil Pressure	____ Oil Pressure	____ Oil Pressure

5. On the back of this page, draw the optimum display format that you would want to use for monitoring and controlling speed, water temperature, fuel level and oil pressure.

Thank you for your participation.

Distribution List

OSD

Dr. Earl Alluisi
Office of the Deputy Under Secretary of Defense
OUSDRE (E&LS)
Pentagon, Room 3D129
Washington, D.C. 20301

Department of the Navy

Dr. L. Chmura
Naval Research Laboratory
Code 5590
Washington, D.C. 20375-5000

Dr. Stanley Collyer
Office of Naval Technology
Code 222
800 N. Quincy Street
Arlington, VA 22217-5000

Perceptual Science Program
Office of Naval Research
Code 1142PS
800 N. Quincy Street
Arlington, VA 22217-5000

Mr. Paul Rau
Code U-33
Naval Surface Weapons Center
White Oak Laboratory
Silver Spring, MD 20903-5000

Other Government Agencies

Defense Technical Information Center
Cameron Station, Bldg. 5
Alexandria, VA 22314 (2 Copies)